1999

Design of three-dimensional digital filters.

Idris S. El-Feghi
University of Windsor

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

Design of Three-Dimensional Digital Filters

by
Idris S. El-Feghi

A Thesis

Submitted to the College of Graduate Studies and Research through the

Department of Electrical and Computer Engineering in Partial Fulfillment

of the Requirements for the Degree of

Master of Applied Science

at the

University of Windsor

Windsor, Ontario, Canada

September 1999.

ABSTRACT

Three-Dimensional (3-D) digital signal processing is becoming increasingly important, and finding applications in diverse areas of current interest, such as computer tomography, geophysics, video and robotics. Three-Dimensional signals offer a significant advantage over 2-D signals which the ability to preserve spacial information. Since almost any detected signal could be contaminated with unwanted noise, 3-D filtering is an important process. In order facilitate 3-D signal processing it is necessary to extend the widely applied 2-D filter design techniques to 3-D. Of the two classes of 3-D filters, namely, recursive and non-recursive , the first one is more efficient in terms of hardware costs. However recursive filters suffer from stability problem and phase linearity.

This thesis deals with the design of 3-D recursive and non-recursive filters by extending some of the 2-D design techniques. In the first part JAVA programming language will be used to implement Simpson'rule for triple integration, 3-D FFT, 3-D window function and 3-D plots of frequency response of the filters.

In the second part linear and non-linear optimization techniques will be used to determine the coefficients of the transfer function of the 3-D digital filter. This is carried out by minimizing an objective function which is a measure of the difference between the magnitude response of the designed and desired 3-D filters. The design methods are used for the general transfer function and for special case of separable denominator and non-separable numerator transfer function.

The design of 3-D filters using optimization techniques requires a considerable a mount of computation time, this can be reduced by imposing certain symmetries on the filters

coefficients, thus reducing the number of independent coefficients and the region of frequency response.

It has been shown that only a 2-D analog transfer function with a 2-D Very Strictly Hurwitz Polynomial-VSHP in its denominator can produce a 2-D stable recursive digital filter upon the application of double bilinear transformation. This thesis also presents the extension of VSHP to the design of 3-D stable digital filter.

To: My Parents,
My Wife Fuzia
and my Children
Khiria, Mohamed and Malik

Acknowledgments

I would like to express my sincere gratitude to both of my co-supervisors Dr. M.A. Sid-Ahmed and Dr. M. Ahmadi for their encouragement, support and guidance during the course of this research. I would like also to thank the thesis committee members, Dr. J. J. Soltis, Dr. A. Alpas and Dr. N. Biswas for their comments and advices.

Finally my warmest acknowledgments to my parents, my wife for there constant love and support and my children Khiria, Mohamed and Malik for their understanding and patience.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE

INTRODUCTION

1.1 Introduction

The last two decades have witnessed tremendous advances in digital system technology with a drop in cost of hardware implementations. Each year, as integrated circuits become faster, cheaper and more compacted, solution to many problems of digital signal processing application have become feasible which stimulate further developments in the theory.

Before 1975 research was confined into one-dimensional (1-D) system, and since 1975 the use of high speed computers have increased with a drop in hardware cost which made the solution of 2-D and 3-D systems problems feasible.

A signal is any medium of conveying information. Digital signal processing is concerned with the representation and the manipulation of digital signals. The processing of the signal represented in the form of sequence of numbers by altering certain characteristics of the signal. The manipulation of 3-D digital signal is commonly referred to as 3-D signal processing. A dimension could mean any physical domain in which the signal is defined. Time, frequency and space are example of such a domain and any combination of the three is allowed. Three-dimensional (3-D) signal processing has recently emerged as an essential tool in many applications such as computer tomography[1][2], geophysics[3], computer vision[4], and video[5][6]. Three-dimensional signals offer the same advantages of 2-D signals plus one more added advantage which is the ability to preserve the spatial information[7].

1.2 THREE-DIMENSIONAL DIGITAL FILTERS

A 3-D digital filter is a computational algorithm that transforms a 3-D input sequence of numbers to a 3-D output sequence of numbers according to a specified rules, hence yielding some desired modifications to the characteristics of the input signal.

Three-dimensional digital filters can be used in such as the removal of moving object along a given path at given speed[5] and in increasing the frame rate of television images.

In some 3-D signal processing applications, such as in computer tomography, all three dimensions are space and must be treated equally. Digital filters are divided into two categories, namely, non-recursive and recursive filters.

A class of three-dimensional filters are defined by linear time-invarient function in z_1, z_2 and z_3. Here z_1 is considered as the pixel delay, z_2 is the line delay and z_3 as the frame delay[8]. Figure 1.1 shows a sequence of images represented by $f(x,y,t)$. A three dimensional z-transform of $f(x,y,t)$ would yield $f(z_1, z_1, z_3)$ as depicted in figure 1.1.

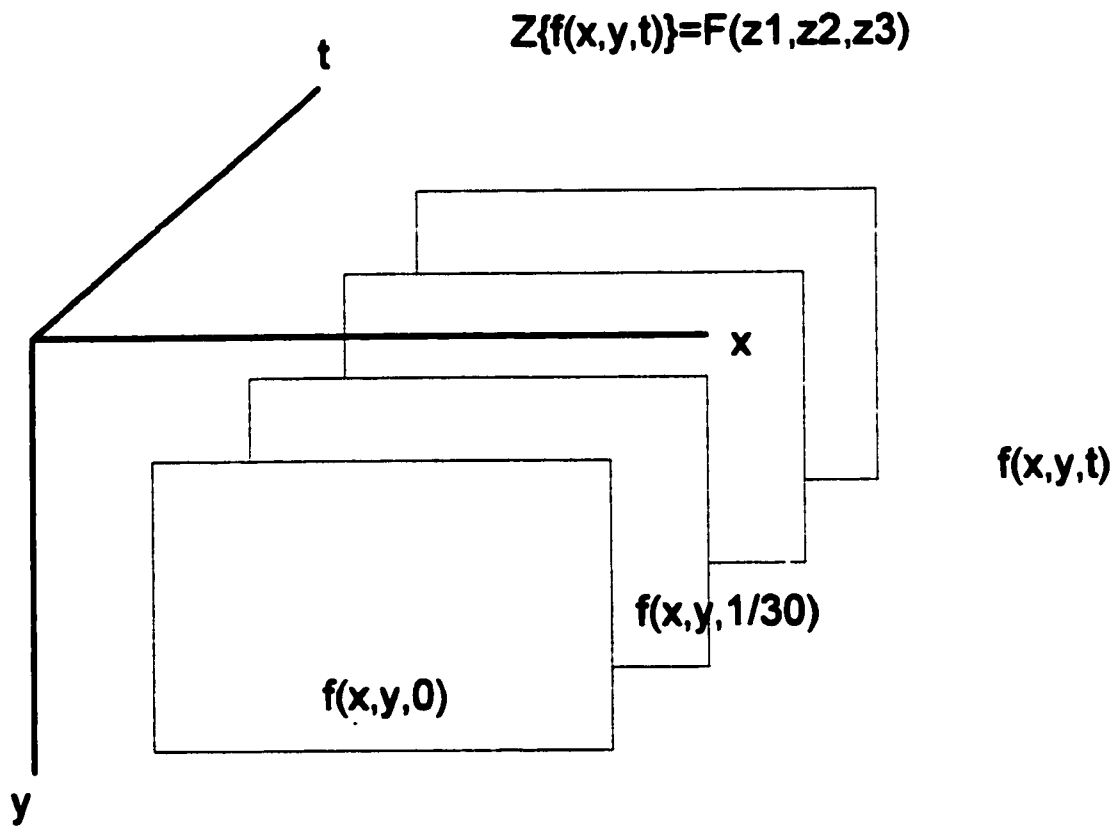

Figure 1.1 Representaion for "Moving " images

1.2.1 NON-RECURSIVE FILTERS

In non-recursive filters the output sequence is a weighted sum of the input sequence over a number of preceding samples. Thus the output *y(l,m,n)* can be written as a function of the input sequence *x(l,m,n)* which is *f(x,y,t)* in fig. 1.1 as

$$y(l,m,n) = \sum_{i_1=-\infty}^{\infty} \sum_{i_2=\infty}^{\infty} \sum_{i_3=-\infty}^{\infty} h(i_1,i_2,i_3)x(l-i_1,m-i_2,n-i_3)$$

(1.1) For a filter to be of finite order and causal its output depends on the present and previous input/output values Eq(1.1) is written as

$$y(l,m,n) = \sum_{i_1=0}^{N-1} \sum_{i_2=0}^{N-1} \sum_{i_3=0}^{N-1} h(i_1,i_2,i_3)x(l-i_1,m-i_2,n-i_3) \qquad (1.2)$$

Where $h(i_1,i_2,i_3)$ is the impulse response of the filter and N is the order of the filter. The transfer function of a 3-D Non-recursive filter is written as

$$H(z_1,z_2,z_3) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} a(i,j,k) z_1^{-i} z_2^{-j} z_3^{-k} \qquad (1.3)$$

Since the impulse response of non-recursive filters is of finite duration they are called Finite Impulse Response or FIR filters.

1.2.2 RECURSIVE FILTERS

A 3-D recursive filter can be characterized by its difference equation as

$$y(l,m,n)=\sum_{i_1=0}^{N-1}\sum_{i_2=0}^{N-1}\sum_{i_3=0}^{N-1}a(i_1,i_2,i_3)x(l-i_1,m-i_2,n-i_3)-\sum_{\substack{i_1=0\\ i_1+i_2+i_3\neq 0}}^{N-1}\sum_{i_2=0}^{N-1}\sum_{i_3=0}^{N-1}b(i_1,i_2,i_3)y(l-i_1,m-i_2,n-i_3) \quad (1.4)$$

or by its transfer function as

$$H(z_1,z_2,z_3)=\frac{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\sum_{k=0}^{N-1}a(i,j,k)z_1^{-i}z_2^{-j}z_3^{-k}}{\sum_{\substack{i=0\\ i+j+k\neq 0}}^{N-1}\sum_{j=0}^{N-1}\sum_{k=0}^{N-1}b(i,j,k)z_1^{-i}z_2^{-j}z_3^{-k}}=\frac{A(z_1,z_2,z_3)}{B(z_1,z_2,z_3)} \quad (1.5)$$

Since the impulse response of a recursive filter is of infinite duration they are called Infinite Impulse Response filters or IIR filters.

Because the output of IIR filters is a function of the present and past input as well as the past output, the output can become increasingly large despite the size of the input, as a result these filters can become unstable. A more detailed stability definition is presented is section 1.4.

1.2.3 SUB-CLASS OF 3-D IIR FILTERS

SEPARABLE DENOMINATOR NON -SEPARABLE NUMERATOR 3-D IIR FILTERS

The transfer function of this class of filters is

$$H(z_1,z_2,z_3)=\frac{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\sum_{k=0}^{N-1}a(i,j,k)z_1^{-i}z_2^{-j}z_3^{-k}}{\left(\sum_{i=0}^{N-1}b(i)z_1^{-i}\right)\left(\sum_{j=0}^{N-1}b(j)z_2^{-j}\right)\left(\sum_{k=0}^{N-1}b(k)z_3^{-k}\right)} \tag{1.6}$$

The main advantages of this type of filters is the following:

1-Stability problem is reduced to 1-D filter stability which can be easily checked.

2-The number of independent coefficients in the denominator is reduced from N^3 to 3N by the use of the different symmetries.

1.3 SYMMETRIES

It is common in many applications that the values of squared magnitude of the frequency response of the desired filter response at different points are interrelated in a certain ways known as symmetries. It is highly desired to use these symmetries in the design so that the number of independent coefficients and the region of frequency domain is reduced [9,10].

The frequency response of a 3-D FIR filter is described by its transfer function

$$H(z_1,z_2,z_3)=A(z_1,z_2,z_3) \tag{1.7}$$

In case of IIR filters the frequency response is a rational function described by

$$H(z_1,z_2,z_3)=\frac{A(z_1,z_2,z_3)}{B(z_1,z_2,z_3)} \tag{1.8}$$

Where $A(z_1,z_2,z_3)$ and $B(z_1,z_2,z_3)$ are polynomials of 3 variables z_1, z_2 and z_3 of the form

$$A(z_1,z_2,z_3)=\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\sum_{k=0}^{N-1}a(i,j,k)z_1^{-i}z_2^{-j}z_3^{-k} \tag{1.9}$$

$$B(z_1,z_2,z_3)=\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\sum_{k=0}^{N-1}b(i,j,k)z_1^{-i}z_2^{-j}z_3^{-k} \tag{1.10}$$

The squared magnitude of the frequency response is defined as[9]

$$F(\omega_1,\omega_2,\omega_3)=H(z_1,z_2,z_3).H(z_1^{-1},z_2^{-1},z_3^{-1})\Big|_{z_1=e^{-j\omega_1},z_2=e^{-j\omega_2},z_3=e^{-j\omega_3}} \tag{1.11}$$

The symmetries in FIR filters are essentially syemmetries of the coefficients of the 3-D polynomial. Asufficient condition for a 3-D IIR filtes to possess certain symmety is that the numerator has to have coefficients syemmetry as FIR filters while the denominator has to be separable.

1.3.1 CUBIC SYMMETRY:

The cubic symmetry is defined in [9] as

$$\begin{aligned} F(\omega_1,\omega_2,\omega_3) &= F(-\omega_1,-\omega_2,-\omega_3) = F(-\omega_1,-\omega_2,\omega_3) = F(-\omega_1,\omega_2,-\omega_3) \\ &= F(-\omega_1,\omega_2,\omega_3) = F(\omega_1,-\omega_2,-\omega_3) = F(\omega_1,-\omega_2,\omega_3) \\ &= F(\omega_1,\omega_2,-\omega_3) \end{aligned} \tag{1.12}$$

It is satisfied iff

$$\begin{aligned} H(z_1,z_2,z_3)H(z_1^{-1},z_2^{-1},z_3^{-1}) &= H(z_1,z_2^{-1},z_3)H(z_1^{-1},z_2,z_3^{-1}) \\ &= H(z_1^{-1},z_2^{-1},z_3)H(z_1,z_2,z_3^{-1}) \\ &= H(z_1^{-1},z_2,z_3)H(z_1,z_2,z_3^{-1}) \end{aligned} \tag{1.13a}$$

Using cubic symmetry 3-D FIR filters the coefficients of the transfer function must be as

$$\begin{aligned} a_{i,j,k} &= a_{i,-j,k} = a_{i,j,-k} = a_{i,-j,-k} = a_{-i,j,k} = a_{-i,-j,k} = a_{-i,j,-k} \\ &= a_{-i,-j,-k} \end{aligned} \tag{1.13b}$$

The filter transfer function could be written as

$$H(z_1,z_2,z_3) = \sum_{i=0}^{\frac{N-1}{2}} \sum_{j=0}^{\frac{N-1}{2}} \sum_{k=0}^{\frac{N-1}{2}} \hat{a}(i,j,k)\cos(i\omega_1).\cos(j\omega_2).\cos(k\omega_3) \tag{1.14}$$

Using cubic symmetry the number of independent coefficients of the filter is reduced to about 1/8 of the total number and total region as shown in figures 1.2 and 1.3.

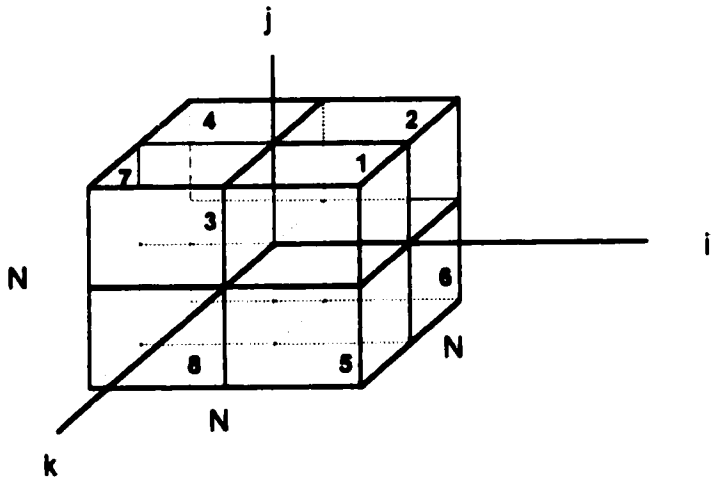

Fig. 1.2 Total frequency region

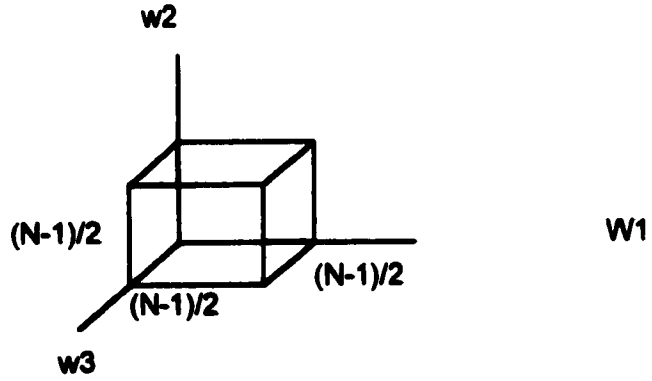

Fig 1.3 Cubic symmetry region

1.3.2 DIAGONAL SYMMETRY

Diagonal symmetry is defined in [9] as

$$F(\omega_1,\omega_2,\omega_3) = F(\omega_1,\omega_3,\omega_3) = F(\omega_2,\omega_1,\omega_3) = F(\omega_3,\omega_2,\omega_1) \tag{1.15}$$

It is satisfied iff

$$\begin{aligned} H(z_1,z_2,z_3)H(z_1^{-1},z_2^{-1},z_3^{-1}) &= H(z_2,z_1,z_3)H(z_2^{-1},z_1^{-1},z_3^{-1}) \\ &= H(z_1,z_3,z_2)H(z_1^{-1},z_3^{-1},z_2^{-1}) \\ &= H(z_3,z_2,z_1)H(z_3^{-1},z_2^{-1},z_1^{-1}) \end{aligned} \tag{1.16a}$$

Using cubic and diagonal symmetries the coefficients FIR filters should have the following additional constraints

$$a_{i,j,k} = a_{j,i,k} \tag{1.16b}$$

transfer function could be written as

$$H(z_1,z_2,z_3) = \sum_{i=0}^{\frac{N-1}{2}} \sum_{j=i}^{\frac{N-1}{2}} \sum_{k=0}^{\frac{N-1}{2}} \hat{a}(i,j,k).\cos(i\omega_1).\cos(j\omega_2).\cos(k\omega_3) \tag{1.17}$$

Using diagonal symmetry the number of independent coefficients and frequency region are reduced by a bout 75% as shown in fig 1.4 .

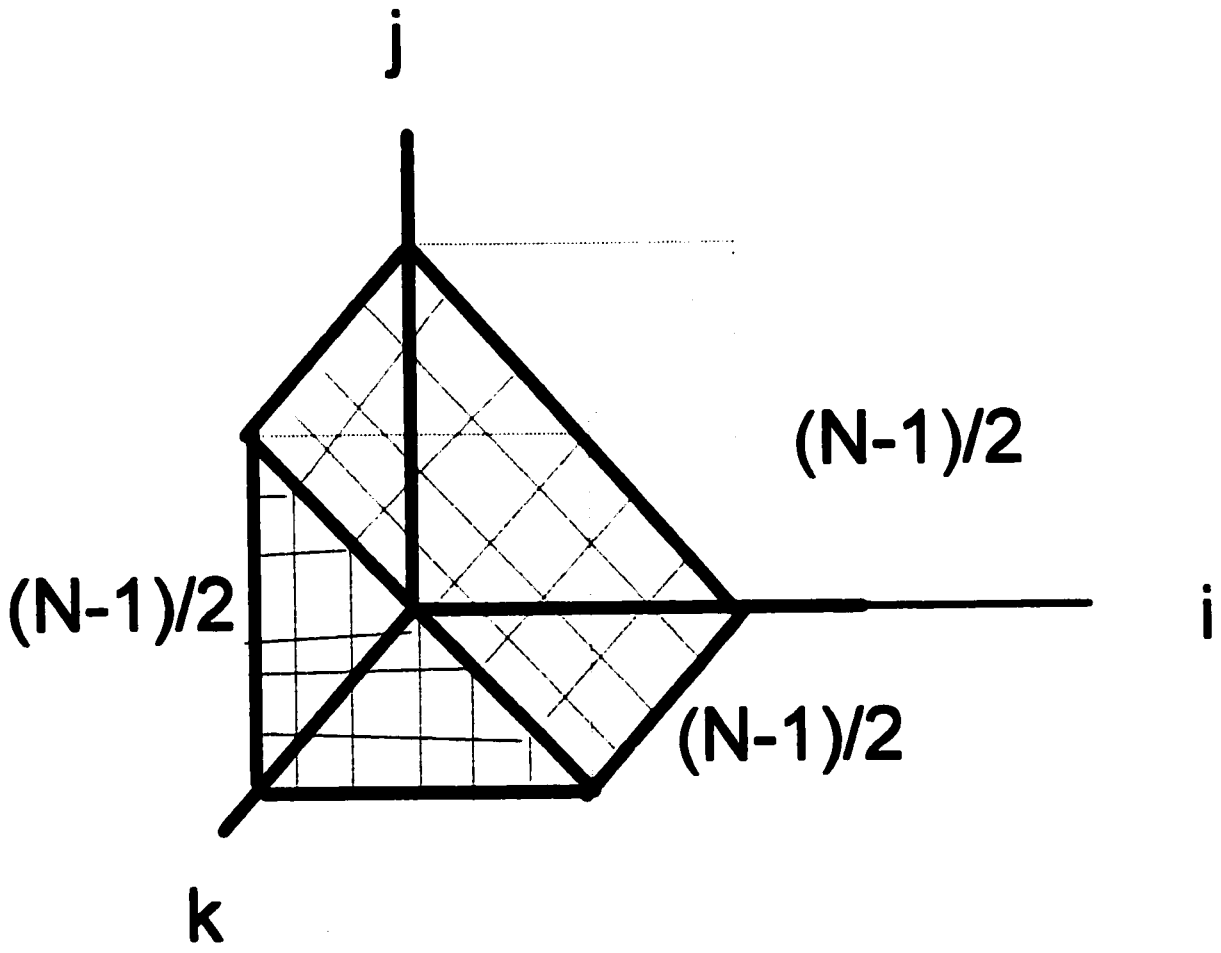

fig. 1.4 Cubic and diagonal symmetry region

1.3.3 16-HYDRAL SYMMETRY

16-Hydral Symmetry is defined in[9] as

$$F(\omega_1,\omega_2,\omega_3) = F(\omega_1,\omega_3,\omega_2) = F(\omega_2,\omega_1,\omega_3) = F(\omega_3,\omega_1,\omega_2) = F(\omega_3,\omega_2,\omega_1) \tag{1.18}$$

This type of symmetry is satisfied iff

$$\begin{aligned} H(z_1,z_2,z_3)H(z_1^{-1},z_2^{-1},z_3^{-1}) &= H(z_1,z_3,z_2)H(z_1^{-1},z_3^{-1},z_2^{-1}) \\ &= H(z_2,z_1,z_3)H(z_2^{-1},z_1^{-1},z_3^{-1}) \\ &= H(z_2,z_3,z_1)H(z_2^{-1},z_3^{-1},z_3^{-1}) \\ &= H(z_3,z_1,z_2)H(z_3^{-1},z_1^{-1},z_2^{-1}) \\ &= H(z_3,z_2,z_1)H(z_3^{-1},z_2^{-1},z_1^{-1}) \end{aligned} \tag{1.19a}$$

Using this symmetry the coefficients of the FIR filter transfer function must have the following additional constraint

$$\begin{aligned} &\text{if } i \neq j \neq k \\ &a_{i,j,k} = a_{i,k,j} = a_{j,i,k} = a_{j,k,i} = a_{k,i,j} = a_{k,j,i} \\ &\text{if } i = j \neq k \\ &a_{i,j,k} = a_{i,k,j} = a_{k,i,j} \\ &\text{if } i = k \neq j \\ &a_{i,j,k} = a_{j,i,k} = a_{i,k,j} \\ &\text{if } j = k \neq i \\ &a_{i,j,k} = a_{j,i,k} = a_{j,k,i} \end{aligned} \tag{1.19b}$$

The transfer function of the filter can be written as

$$H(z_1,z_2,z_3) = \sum_{i=0}^{\frac{N-1}{2}} \sum_{j=i}^{\frac{N-1}{2}} \sum_{k=j}^{\frac{N-1}{2}} \hat{a}(i,j,k).\cos(i\omega_1).\cos(j\omega_2).\cos(k\omega_3) \qquad (1.20)$$

and the frequency region is reduced to that shown in fig. 1.5

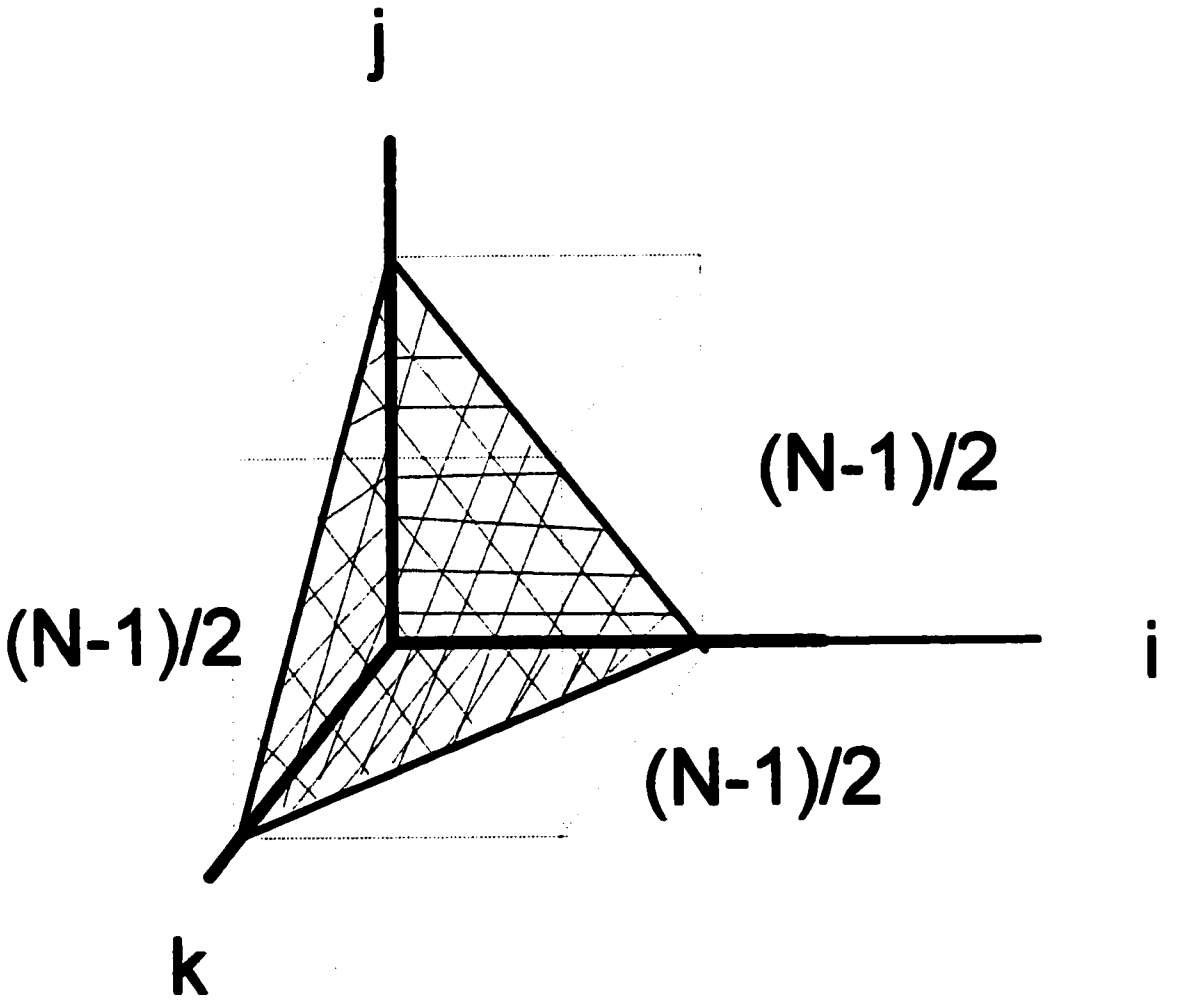

Fig 1.5 16-Hydral symmetry region

1.4 STABILITY

Stability of three-dimensional filters has been subject to research by many researchers [10,11 12,13,14, 15]. An IIR filter is said to be stable if the convolution of its impulse response with an absolutely bounded input sequence will always yield a bounded output. This is the most commonly used definition of stability and also known as Bounded Input Bounded Output (BIBO)[17].

A 3-D absolutely bounded sequence is defined as

$$|x(i,j,k)| \leq P \leq \infty \quad \text{for all } i, j, k \tag{1.21}$$

Where P is a positive number.

A 3-D system is a BIBO and stable if its impulse response is absolutely summable

$$\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\sum_{k=0}^{N-1}|h(i,j,k)| = S < \infty \tag{1.22}$$

Where S is a real finite number

We recall that the output of IIR filter is expressed as the convolution of its impulse response with the input signal as

$$y(l,m,n) = \sum_{i=0}^{\infty}\sum_{j=0}^{\infty}\sum_{k=0}^{\infty}h(i,j,k)x(l-i,m-j,n-k) \tag{1.23}$$

using Schwartz's inequality eq 1.23 can be written as

$$y(l,m,n) \leq \sum_{i=0}^{\infty}\sum_{j=0}^{\infty}\sum_{k=0}^{\infty} |h(i,j,k)\|x(l-i,m-j,n-k)| \tag{1.24}$$

Since $x(i,j,k) <= P$ for all i,j and k the inequality reduces to

$$|y(l,m,n)| \leq P\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}\sum_{k=0}^{\infty}|h(i,j,k)| \tag{1.25}$$

This restricts the impulse response to be absolutely summable and the output array to be bounded provided that the input array is bounded.

Consider the transfer function of 3-D IIR filter

$$H(z_1,z_2,z_3) = \frac{A(z_1,z_2,z_3)}{B(z_1,z_2,z_3)} \tag{1.26}$$

A direct extension of Shank's stability theorem can be stated as follows:

given $B(z_1,z_2,z_3)$ is a polynomial of z_1 ,z_2 and z_3 a necessary and sufficient condition for $H(z_1,z_2,z_3)$ to converge absolutely is

$$B(z_1,z_2,z_3) \neq 0 \quad \text{for} \bigcap_{i=1}^{3}|z_i| \geq 1 \tag{1.27}$$

The above condition suggests that, stable filters must not have any poles outside the unit

sphere. However, testing stability is difficult if not impossible because the fundamental theorem of algebra prevents us from factorizing 3-variable polynomial as a product of the first, second and third order polynomials in z_1, z_2 and z_3.

1.5 COMPARISON BETWEEN FIR AND IIR FILTERS

The choice between FIR and IIR filters depends on the weight of the advantages and disadvantages of each type of filter. In IIR filters the poles can be placed anywhere inside the unit sphere. This degree of freedom permits the designer to meet the specification with low order filter. In the case of FIR filters poles can be placed only at the origin which forces the designer to use higher order filter compared to IIR filters for the same specifications. In term of hardware, higher order is translated to higher cost.

Although IIR filters enjoy the low cost advantage they suffer from the disadvantages of stability which are inherent in FIR filters. Since the impulse response of FIR filters are always absolutely summable they are always stable.

FIR filters they provide an exact linear phase while linear phase is hard to come by in IIR filters.

Stability and linear phase are the advantages of FIR filters while low cost, faster filtering and easier implementation are the advantages of IIR filters.

1.6 DESIGN TECHNIQUES FOR 3-D FIR FILTERS

The impulse response of a FIR filter can be obtained from the frequency response through the inverse Fourier transform. The impulse response obtained is of an infinite duration. The most straightforward approach in designing FIR filter is truncating these infinite-duration impulse response sequences. This truncation eliminates the components beyond certain points L, M, N directions. This truncation unfortunately introduces an unwanted oscillations in the passband and the stopband of the filter known as *Gibbs' oscillations.*

Instead of simply truncating the infinite Fourier series, the techniques of windowing seeks to smoothen the *Gibbs' oscillations* by multiplying the impulse response by a time-limited window. Windowing is the simplest technique with the shortest design time, but it gives a suboptimal results.

One of the most popular techniques for the design of 3-D FIR filters is the extension of McClellan transformation to 3-D[17]. The extension of the generalized McClellan technique to 3-D is straight forward[18]. This techniques splits the design problem into two steps . First the transformation coefficients are determined. Second, the coefficients of 1-D FIR filter is determined by one of the design techniques for 1-D FIR filters[19][20]. The main problem with this design method is the existence of a very large number of coefficients. By imposing symmetry the number coefficients is reduced[21]. Although this technique enjoys short design time, the result obtained is suboptimal in the minimax sense.

Another techniques for designing FIR filters is linear programming. Linear programming has

been widely used in the design of 2-D FIR filters[22][23]. By minimizing an objective function which is a measure of the difference between the desired and designed magnitude and phase response of the filter, a set of filter coefficients approximating the desired responses is obtained. Using linear programming techniques, 3-D FIR filters have been successfully designed by Higuchi, M. Ohki and Kawamata[24]. Although linear programming is very flexible and can be used to approximate a wide variety of desired filter shapes, it is comparatively slow and hence the dimension of the filters that can be designed is limited. Extension of many of the existing method of 1-D filter design to 3-D using non-linear programming is a straightforward as in[25].

1.7 DESIGN TECHNIQUES FOR 3-D IIR FILTERS

Three-Dimensional IIR filter have been realized as a parallel and/or cascade arrangements of 1-D filters, each one designed with respect to one of the dimensions z_1, z_2 and z_3 [26].

By cascading three 1-D all-poles filters with a 3-D FIR filter and using linear programming to obtain filter coefficients, 3-D IIR filters are deigned as in[27]. A technique based on cascade arrangement of 2-D rotated filters and results in 3-D spherically symmetric filters is proposed by M. Zervakis and A. Venetsanopulos[28]. Based on the concept of minimal decomposition and balanced model reduction, a technique for designing three-dimensional separable denominator digital filters was developed in [29] In [30],a 3-D separable-denominator digital filter has been characterized by three 1-D linear filter and the local state-space model has been designed through singular value decomposition of Hankel matrix.

Nowrouzian, Ahmadi and King[25] extended the work of Kalman[31], Steiglitz and MoBride[32], Shanks[33],Bertran[34] and Bordner[35] to N-dimensional by approximating discrete and finite space impulse response by the impulse response of a recursive digital filter. Since these techniques minimize the true mean square and are non-linear in nature, the solution does not necessarily convergence to a global minimum and the stability is not always guaranteed.

Wan and Fahmy [36] proposed a new design technique for n-dimensional (N-D) IIR filters, based on the use of suboptimal stepsize to increase the overall computation efficiency. Criteria were developed for suboptimal stepsize to ensure positive definite matrix and to ensure the stability of the transfer function.

Chottera and Jullien [37] have modified the linear programming approach of Rabiner et al.[38] by including the stability constraints in the process of optimization, as well as including the constant group delay response in the design process. In [39] the approach of [37] is extended to 3-D. In the approach outlined in[37][39] stability constraints on the real part of the numerator of the filter transfer function are necessary but not are sufficient conditions for filter stability.

In most of the approaches outlined previously it is hard to test stability of the designed filter. In addition, there is a deficiency of a solid theoretical background and procedure for stabilizing 3-D filters. A straightforward and successful method for designing a 3-D digital filter is to assign a stable 3-variable polynomial in the denominator of an analog transfer function. Then, by application of triple bilinear transformation, the 3-D digital filter transfer

function can be formed. Goodman[40] showed that not all analog filters will yield a stable digital filters upon the application of bilinear transformation.

Rajan al.[41] Showed that only a special class of analog filters can be transformed to a stable digital filters using bilinear transformation. This special class filters have a Very Strictly Hurwitz Polynomial (VSHP) denominators. VSHP has been widely used in the design of 2-D digital filters as in[42,43,44]. In [45] 3-variable passive RLC is used to design planer filters an in [46] 3-variable VSHP is used to design stable 3-D IIR filters with arbitrary magnitude response. Although the use of 3-variable VSHP results in a stable filter, it is very hard to control the shape of the magnitude response due to the nature of VSHP . The use of non-linear programming in obtaining filter coefficients limits the use of the method to design of low order filters.

1.8 THESIS ORGANIZATION

This thesis is divided into six chapters. In the second chapter we design a 3-D FIR filters by determining the impulse response from the frequency response. Since analytical expression of the impulse response is not always easy to obtain, numerical integration will be used to obtain the impulse response of the filter. Simpson's rule for triple integration is utilized.

Chapter three starts with brief introduction to DFT and FFT followed by their extension to 3-D.Three-dimensional FFT algorithm is developed next and extension of the design techniques using FFT and windows is presented for the design of FIR filters.

Chapter four investigates the extension of Shank's method for the design of 3-D near linear phase IIR filter. Example to demonstrate the results is presented at the end of each chapter.

Chapter five introduces the use of linear programming in the design of 3-D FIR filters by minimizing a cost function which is a measure of the difference between the ideal frequency and the approximated frequency. Examples are presented using different types of symmetries to reduce the number of coefficients and the frequency regions. Linear programming is also used to design a stable, near linear phase IIR filter. Since the approximation procedure is based on linear programming, the constraints that are to be used for stable filter design are required to be linear. Therefore various linear stability constraints are incorporated to the design of stable 3-D IIR filters. Chapter five also presents the design of subclass 3-D IIR filters with separable denominator and non-separable numerator.

In chapter six we present the use of Very Strictly Hurwitz Polynomial (VSHP) to design stable 3-D IIR filters.

The final chapter presents the conclusion and the results derived in the this thesis.

CHAPTER TWO

DESIGN OF 3-D DIGITAL FIR FILTERS USING NUMERICAL INTEGRATION

2.1 INTRODUCTION

This chapter deals with the design of 3-D non-recursive digital filters satisfying prescribed magnitude specifications. The method presented here is straightforward and requires minimal amount of computations. Unfortunately, the designs obtained are sub-optimal. This method extends to 3-D the 2-D approach presented by M.A. Sid-Ahmed in [8]

By designing 3-D filter we mean calculation of the coefficients of the transfer function or the impulse response in such a way that the magnitude response of the designed filter approximate the desired magnitude response. In the next section we present an efficient algorithm for obtaining the impulse response of the filter from the frequency response. Some examples to demonstrate the use of this method are presented

2.2 Determining the impulse response from the frequency response

The frequency response of a 3-D FIR filter is given by

$$H(\omega_1,\omega_2,\omega_3) = \sum_{i_1=0}^{N}\sum_{i_2=0}^{N}\sum_{i_3=0}^{N} h(i_1,i_2,i_3)e^{-j(i_1\omega_1+i_2\omega_2+i_3\omega_3)} \qquad (2.1)$$

$$h(i_1,i_2,i_3) = \frac{1}{8\pi^3}\int_{-\pi}^{\pi}\int_{-\pi}^{\pi}\int_{-\pi}^{\pi} H(\omega_1,\omega_2,\omega_3)e^{j(\omega_1+\omega_2+\omega_3)}d\omega_1 d\omega_2 d\omega_3 \qquad (2.2)$$

This means that the impulse response of the filter can be determined from the frequency response through eq(2.2).

Design example 2.1

Given the frequency response described by

$$H(\omega_1,\omega_2,\omega_3) = \begin{cases} 1 & a \le |\omega_1| \le \pi,\ b \le |\omega_2| \le \pi,\ c \le |\omega_3| \le \pi \\ 0 & \text{otherwise} \end{cases}$$

Determine the impulse response

Solution . From eq(2.2) the impulse response can be written as

$$h(i_1,i_2,i_3) = \frac{1}{8\pi^3}\int_{-a}^{a}\int_{-b}^{b}\int_{-c}^{c} e^{j(i_1\omega_1+i_2\omega_2+i_3\omega_3)}d\omega_1 d\omega_2 d\omega_3$$

$$h(i_1,i_2,i_3) = \frac{1}{2\pi}\int_{-a}^{a} e^{ji_1\omega_1}d\omega_1 \frac{1}{2\pi}\int_{-b}^{b} e^{ji_2\omega_2}d\omega_2 \frac{1}{2\pi}\int_{-c}^{c} e^{ji_3\omega_3}d\omega_3$$

$$= \frac{\sin(an_1)}{n_1\pi} \cdot \frac{\sin(bn_2)}{n_2\pi} \cdot \frac{\sin(cn_3)}{n_3\pi}$$

An analytical expression for the impulse response is not always possible . The following example demonstrate the use of numerical integration for the generation of the impulse response.

Example 2.2 Generate the impulse response of 3-D FIR low-pass butterworth filter whose frequency response is given by

$$H(\omega_1,\omega_2,\omega_3) = \frac{1}{1+\frac{\sqrt{2}-1}{D_o}R^2(\omega_1,\omega_2,\omega_3)}$$

Where $-\pi \le \omega_i \le \pi$ i=1,2,3

$R(\omega_1,\omega_2,\omega_3) = \sqrt{\omega_1^2+\omega_2^2+\omega_3^2}$ $D_o = 0.3\pi$ The impulse response is taken over a cube of size 7×7×7.

Since $H(\omega_1,\omega_2,\omega_3)$ is real and has a spherical symmetry its impulse response $h(n_1,n_2,n_3)$ can be written as

$$h(n_1,n_2,n_3)=\frac{1}{8\pi^3}\int_{-\pi}^{\pi}\int_{-\pi}^{\pi}\int_{-\pi}^{\pi}H(\omega_1,\omega_2,\omega_3)\cos(n_1\omega_1+n_2\omega_2+n_3\omega_3)d\omega_1 d\omega_2 d\omega_3$$

Which can be further reduced to

$$h(n_1,n_2,n_3)=\frac{1}{\pi^3}\int_{-\pi}^{\pi}\cos(n_1\omega_1).\left[\int_{-\pi}^{\pi}\cos(n_2\omega_2)\left[\int_{-\pi}^{\pi}H(\omega_1\omega_2,\omega_3)\cos(n_3\omega_3)d\omega_3\right]d\omega_2\right]d\omega_1$$

The solution to this problem will be carried out numerically through a program written in the JAVA programming language. JAVA has been widely adapted in a wide variety of applications because of its speed and flexibility. Before we develop the program , we need to investigate a method for triple integration. The Simpson 's rule for triple integration will be developed next.

Simpson's rule for numerical integration is given by[47]

$$I=\int_{x_0}^{x_m}f(x)dx$$

$$\approx\frac{\Delta x}{3}\left[f(x_0)+4\sum_{\substack{i=1,3,5..\\ odd}}^{m-1}f(x_i)+2\sum_{\substack{i=2,4,6..\\ even}}^{m-2}f(x_i)+f(x_m)\right]$$

for double integration

$$I = \int_{x_0}^{x_m}\int_{y_0}^{y_n} f(x,y)\,dx\,dy$$

$$\approx \frac{\Delta x}{3}\int_{y_0}^{y_n}\left[f(x_0,y) + 4\sum_{\substack{i=1,3,5..\\ odd}}^{m-1} f(x_i,y) + 2\sum_{\substack{i=2,4,6..\\ even}}^{m-2} f(x_i,y) + f(x_m,y)\right]dy$$

This can be written as

$$I \approx \frac{\Delta x}{3}\left(\int_{y_0}^{y_n}[1\ 4\ 3\ 4\ 2\ \ldots 4\ 2\ 4\ 1]\begin{bmatrix} f(x_0,y)\\ f(x_1,y)\\ f(x_2,y)\\ .\\ .\\ .\\ f(x_m,y)\end{bmatrix}\right)dy$$

$\Delta x = \dfrac{x_m - x_0}{m}$ m is an even number

Using Simpson's rule a gain on the previous equation we get

$$I \approx \frac{\Delta x}{3}.\frac{\Delta y}{3}.[1\ 4\ 2\ 4 \ldots 2\ 4\ 1]\begin{bmatrix} f(x_0,y_0) & f(x_0,y_1) & f(x_0,y_2) & \ldots & f(x_0,y_n) \\ f(x_1,y_0) & f(x_1,y_1) & f(x_1,y_2) & \ldots & f(x_1,y_n) \\ f(x_2,y_0) & f(x_2,y_1) & f(x_2,y_2) & \ldots & f(x_2,y_n) \\ \ldots & \ldots & \ldots & & \ldots \\ \ldots & \ldots & \ldots & & \ldots \\ \ldots & \ldots & \ldots & & \ldots \\ f(x_m,y_0) & f(x_m,y_1) & f(x_m,y_2) & \ldots & f(x_m,y_n) \end{bmatrix}\begin{bmatrix} 1 \\ 4 \\ 2 \\ . \\ . \\ . \\ 2 \\ 4 \\ 1 \end{bmatrix}$$

for triple integration

$$I \approx \frac{\Delta x}{3}.\frac{\Delta y}{3}.\int_{z_0}^{z_t}([1\ 4\ 2\ 4 \ldots 2\ 4\ 1]\begin{bmatrix} f(x_0,y_0,z) & f(x_0,y_1,z) & f(x_0,y_2,z) & \ldots & f(x_0,y_n,z) \\ f(x_1,y_0,z) & f(x_1,y_1,z) & f(x_1,y_2,z) & \ldots & f(x_1,y_n,z) \\ f(x_2,y_0,z) & f(x_2,y_1,z) & f(x_2,y_2,z) & \ldots & f(x_2,y_n,z) \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ f(x_m,y_0,z) & f(x_m,y_1,z) & f(x_m,y_2,z) & \ldots & f(x_m,y_n,z) \end{bmatrix}\begin{bmatrix} 1 \\ 4 \\ 2 \\ . \\ . \\ . \\ 2 \\ 4 \\ 1 \end{bmatrix})dz$$

Applying Simpson's rule again we get

$$I \approx \frac{\Delta x}{3}.\frac{\Delta y}{3}.\frac{\Delta z}{3}[1\,4\,2\,4\ldots2\,4\,1]\begin{bmatrix} f(x_0,y_0,z_0) & f(x_0,y_0,z_1) & f(x_0,y_0,z_2) & \ldots & f(x_0,y_0,z_l) \\ f(x_0,y_1,z_0) & f(x_0,y_1,z_1) & f(x_0,y_1,z_2) & \ldots & f(x_0,y_1,z_l) \\ f(x_0,y_2,z_0) & f(x_0,y_2,z_1) & f(x_0,y_2,z_2) & \ldots & f(x_0,y_2,z_l) \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ f(x_0,y_n,z_0) & f(x_0,y_n,z_1) & f(x_0,y_n,z_2) & \ldots & f(x_0,y_n,z_l) \\ f(x_1,y_0,z_0) & f(x_1,y_0,z_1) & f(x_1,y_0,z_2) & \ldots & f(x_1,y_0,z_l) \\ f(x_1,y_1,z_0) & f(x_1,y_1,z_1) & f(x_1,y_1,z_2) & \ldots & f(x_1,y_1,z_l) \\ \ldots & \ldots & \ldots & \ldots & \\ f(x_1,y_n,z_0) & f(x_1,y_n,z_1) & f(x_0,y_n,z_2) & \ldots & f(x_0,y_n,z_l) \\ f(x_2,y_0,z_0) & f(x_2,y_0,z_1) & f(x_2,y_0,z_2) & \ldots & f(x_2,y_0,z_l) \\ \ldots & \ldots & \ldots & \ldots & \\ f(x_m,y_n,z_0) & f(x_m,y_n,z_1) & f(x_m,y_n,z_2) & \ldots & f(x_m,y_n,z_l) \end{bmatrix}\begin{bmatrix} 2 \\ 4 \\ 2 \\ . \\ 2 \\ 4 \\ 2 \\ . \\ 4 \\ 2 \\ 1 \end{bmatrix}$$

where $\Delta x = \frac{x_m - x_0}{n}$, $\Delta y = \frac{y_n - y_0}{m}$, $\Delta z = \frac{z_l - z_0}{l}$

m, n and l are even numbers

If $H(\omega_1,\omega_2,\omega_3)$ is real and have spherical symmetry, then the impulse response can be obtained from the triple integration

$$h(n_1,n_2,n_3) = \int_0^{\pi}\int_0^{\pi}\int_0^{\pi} H(\omega_1,\omega_2,\omega_3)\cos(\omega_1 n_1)\cos(\omega_{21} n_2)\cos(\omega_3 n_3)\,d\omega_1 d\omega_2 d\omega_3$$

Using the above formula a JAVA program is developed for obtaining the impulse response of the desired filter . The source code is listed in appendix A. Table 2.1 shows one cube of the impulse response of a low pass filter calculated over a cube of 7 x 7 x 7 with cut off

frequency of 0.3π .

Table 2.1 One cube of impulse response of a low-pass filter with a cut-off $0.3\ \pi$

a(0,0,0)...a(0,0,3)	0.05615156	0.01891741	0.0049776877	0.0037206248
	0.01891741	0.009673654	0.0038169036	0.0021288323
a(0,3,0)...a(0,3,3)	0.0049776877	0.0038169036	0.001980273	0.0010867149
	0.0037206248	0.0021288323	0.0010867149	5.5965374E-4
a(1,0,0)...a(0,0,3)	0.018058738	0.009901635	0.0038502899	0.002170635
	0.009901635	0.0063489187	0.003037307	0.001640312
a(1,3,0)...a(1,3,3)	0.0038502899	0.003037307	0.0016831604	9.7303494E-4
	0.002170635	0.001640312	9.7303494E-4	5.14195E-4
a(2,0,0)...a(2,0,3)	0.003939482	0.0034036906	0.0020679892	0.0011425953
	0.0034036906	0.0027215541	0.0017374904	9.871129E-4
a(2,3,0)...a(2,3,3)	0.0020679892	0.0017374904	0.0011004058	6.969355E-4
	0.0011425953	9.871129E-4	6.969355E-4	3.8267806E-4
a(3,0,0)...a(3,0,3)	0.0022755347	0.0015640485	0.0010027762	6.5285276E-4
	0.0015640485	0.0012391299	8.815154E-4	5.6559156E-4
a(3,3,0)...a(3,3,3)	0.0010027762	8.815154E-4	6.2078796E-4	4.199967E-4
	6.5285276E-4	5.6559156E-4	4.199967E-4	2.5841172E-4

Fig. 2.1,2.2,2.3 show 3-D plots of the frequency response of the filter at different values of ω_3 .

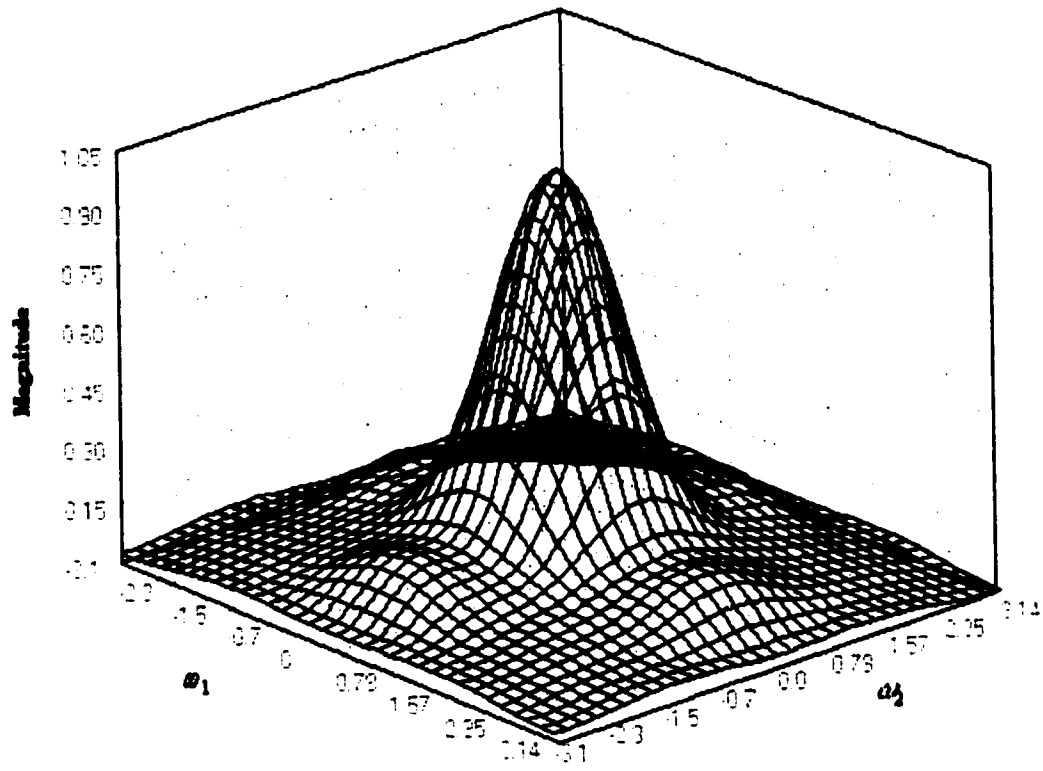

Figure 2.1 frequency response of 7 x 7 x 7 low-pass filter with $\omega_c = 0.3\pi$ at $\omega_3 = 0$

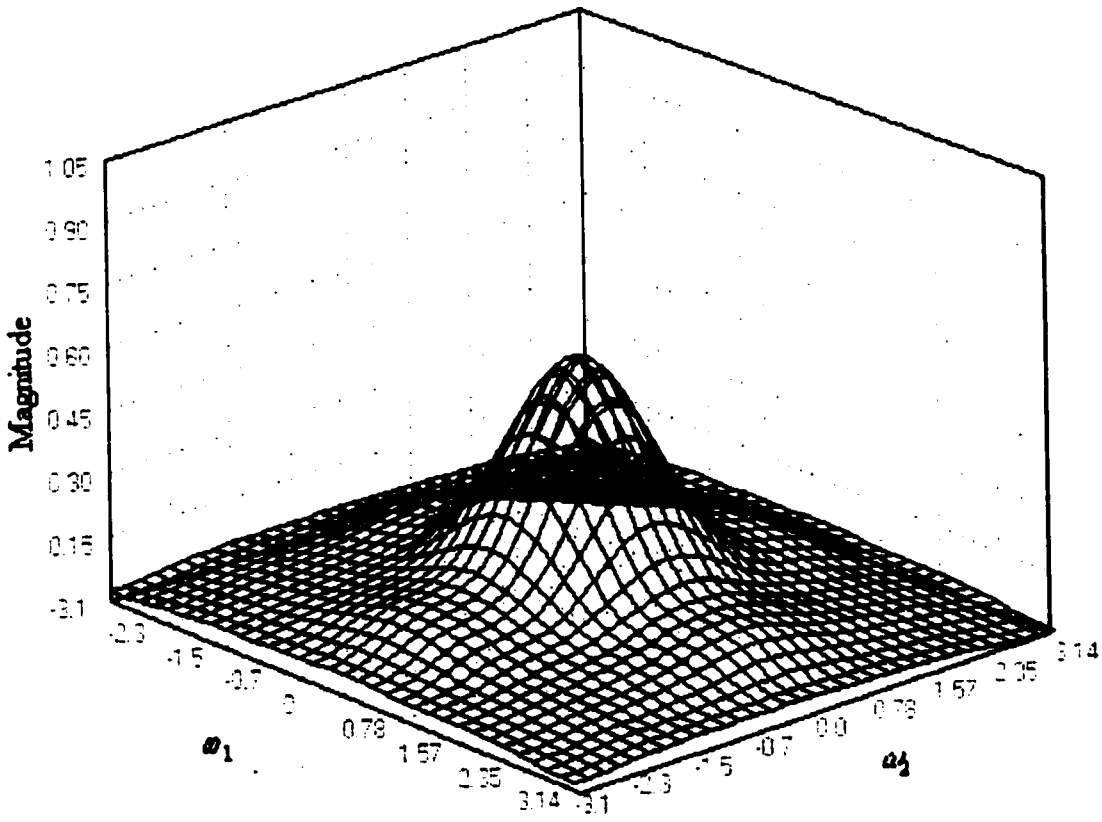

Figure 2.2 frequency response of 7 x 7 x 7 low-pass filter with $\omega_c = 0.3\pi$ at $\omega_3 = 1.178$

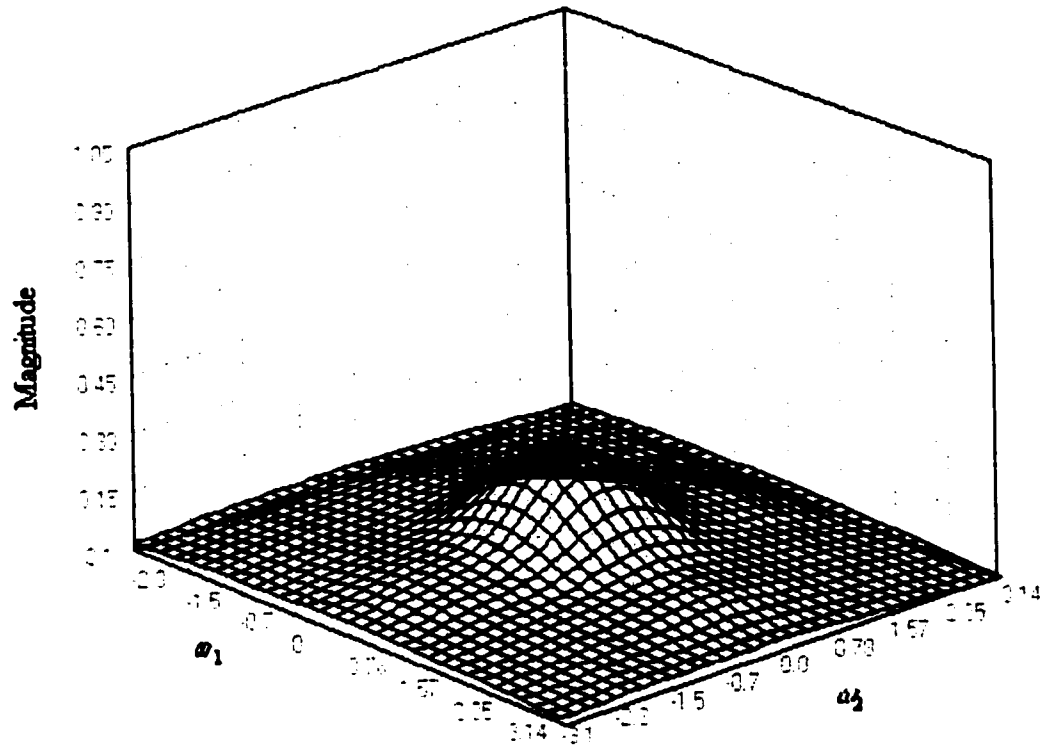

Figure 2.3 frequency response of 7 x 7 x 7 low-pass filter with $\omega_c = 0.3\pi$ at $\omega_3 = 2.36$

Example 2.2 Generate the impulse response of spherically symmetric 3-D high-pass butterworth filter whose frequency response is described by

$$H(\omega_1,\omega_2,\omega_3) = \frac{1}{1+\frac{\sqrt{2}-1}{D^2}R^2(\omega_1,\omega_2,\omega_3)}$$

where $-\pi \le \omega_1 \le \pi,\ -\pi \le \omega_2 \le \pi,\ -\pi \le \omega_3 \le \pi$

and $R^2(\omega_1,\omega_2,\omega_3) = \sqrt{\omega_1^2+\omega_2^2+\omega_3^2}$

Table 2.2 show one cube of the impulse response of the designed filter taken over a size of 9x9x9. Fig2.4,2.5 and 2.6 show 3-D plots of the magnitude response at different values of ω_3

Table 2.2 One cube of impulse response of the designed filter over a window of 9 X 9X9

$a_{0,0,0}\cdots a_{0,0,4}$	0.943779	0.005874305	0.0032881538	0.02107109	0.0063403808
	0.005874305	-0.01008868475	-0.003954634	-0.0025420263	-0.0013776107
	0.0032881538	-0.003954634	-0.0019802733	-0.0010867162	-6.6369114E-4
$\ddot{a}_{0,4,0}\cdots a_{0,4,4}$	0.02107109	-0.0025420263	-0.0010867162	-9.7284775E-4	-4.958597E-4
	0.0063403808	-0.0013776107	-6.6369114E-4	-4.958597E-4	-1.8700044E-4
	-0.009795958	-0.008696773	-0.00378200027	-0.0019657721	-0.0011889251
	-0.008696773	-0.0063489187	-0.003037307	-0.001640312	-9.7543834E-4
	-0.00378200027	-0.003037307	-0.0016843178	-9.765072E-4	-5.943431E-4
	-0.0019657721	-0.001640312	-9.765072E-4	-5.14195E-4	-3.306461E-4
	-0.0011889251	-9.7543834E-4	-5.943431E-4	-3.306461E-4	-1.8552601E-4
	0.0044632484	-0.0031953582	-0.0019985456	-9.342628E-4	-6.241573E-4
	-0.0031953582	-0.0031347494	-0.0018752211	-0.0014003081	-7.4249494E-4
	-0.0019985456	-0.0018752211	-9.615177E-4	-7.8026807E-4	-2.9354283E-4
	-9.342628E-4	-0.0014003081	-7.8026807E-4	-7.958733E-4	-3.935744E-4
	-6.241573E-4	-7.4249494E-4	-2.9354283E-4	-3.935744E-4	-4.7343256E-6
	0.0059872447	-0.0013591858	-9.3448913E-4	-4.4799008E-4	-3.3804006E-4
	-0.0013591858	-0.0012391299	-8.815154E-4	-5.6559136E-4	-3.6051206E-4
	-9.3448913E-4	-8.815154E-4	-5.2194536E-4	-4.2346897E-4	-2.752086E-4
	-4.4799008E-4	-5.6559136E-4	-4.2346897E-4	-2.5841172E-4	-1.7734169E-4
	-3.3804006E-4	-3.6051206E-4	-2.752086E-4	-1.7734169E-4	-1.0611179E-4
$a_{4,0,0}\cdots a_{4,0,4}$	0.007738747	-3.8454533E-4	-4.0286584E-4	-1.1907851E-4	-1.494869E-4
	-3.8454533E-4	-8.834277E-4	-6.857773E-4	-7.000509E-4	-3.356981E-4
	-4.0286584E-4	-6.857773E-4	-1.8786331E-4	1.6951202E-4	-2.675463E-5
$\ddot{a}_{4,4,0}\cdots a_{4,4,4}$	-1.1907851E-4	-7.000509E-4	1.6951202E-4	-5.599417E-4	-2.4641788E-4
	-1.494869E-4	-3.356981E-4	-2.675463E-5	-2.4641788E-4	6.4913E-5

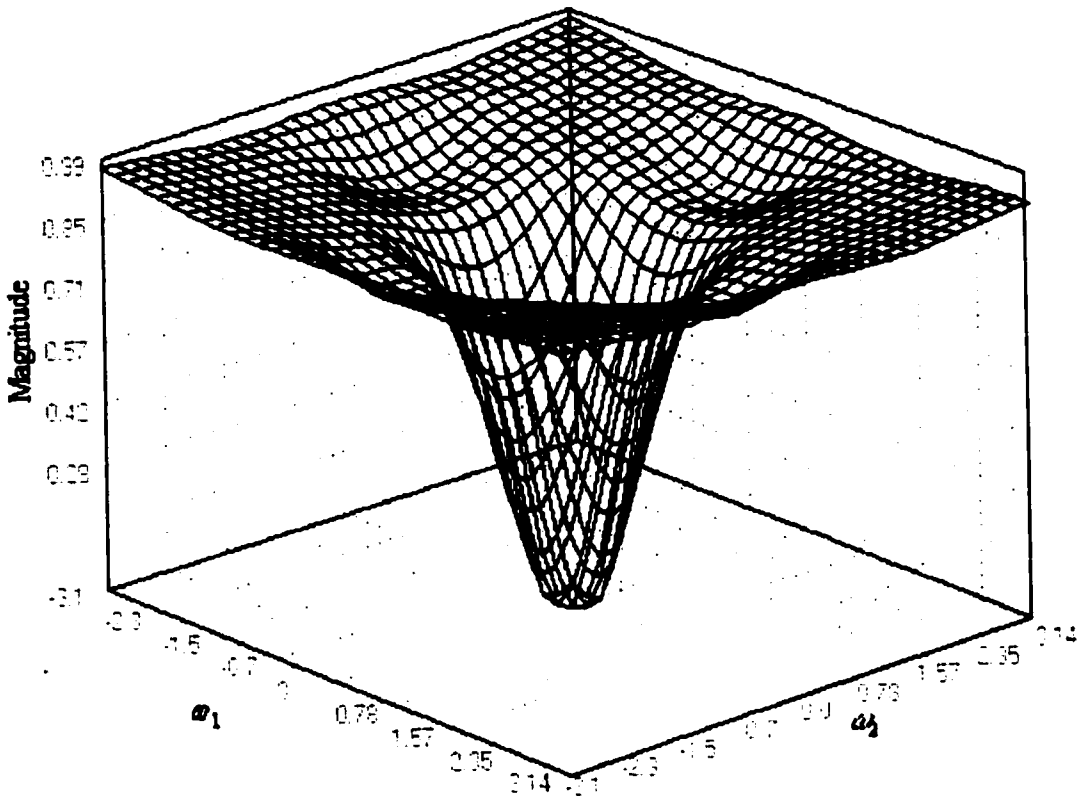

Figure 2.4 frequency response of 9 x 9 x 9 high-pass filter with $\omega_c = 0.3\pi$ at $\omega_3 = 0$

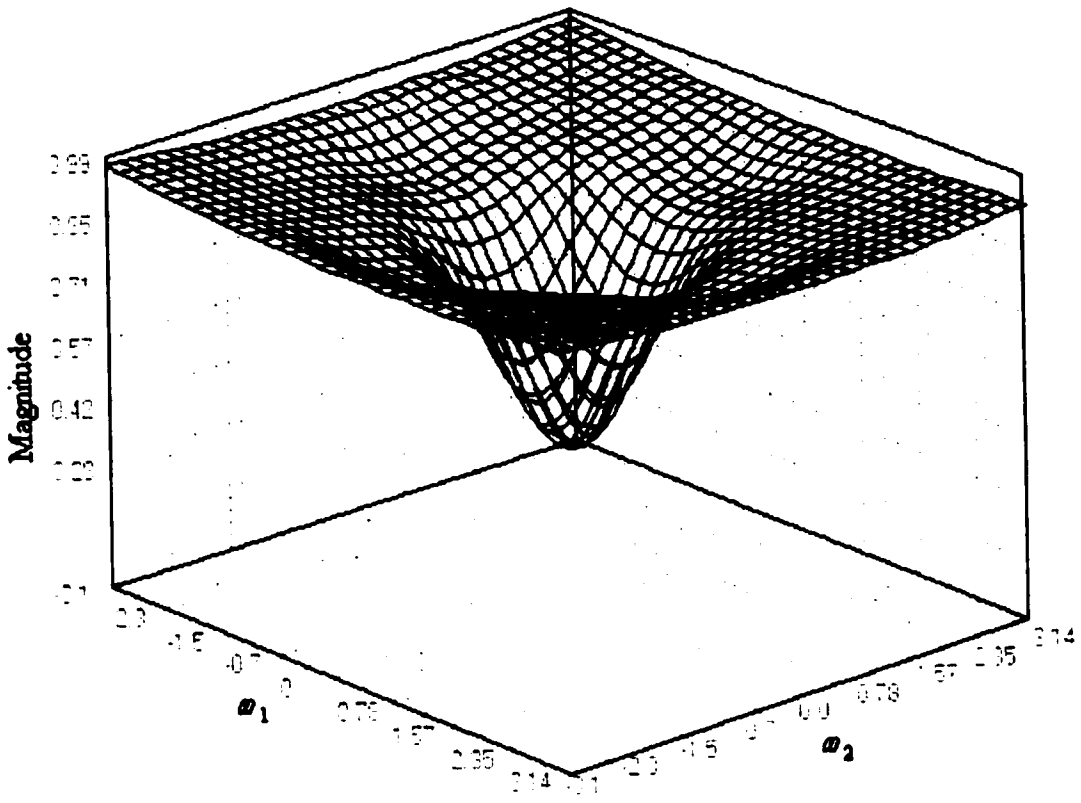

Figure 2.5 frequency response of 9 x 9 x 9 high-pass filter with $\omega_c = 0.3\pi$ at $\omega_3 = 1.178$

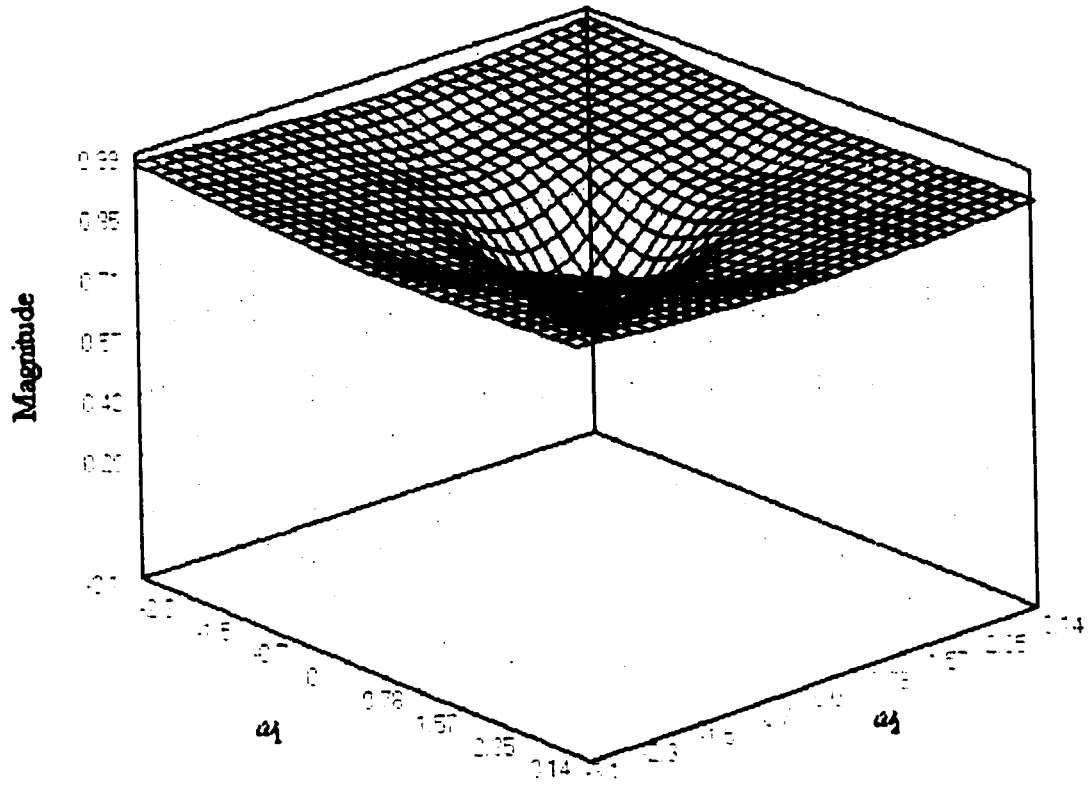

Figure 2.6 frequency response of 9 x 9 x 9 high-pass filter with $\omega_c = 0.3\pi$ at $\omega_3 = 2.16$

CHAPTER THREE

DESIGN OF 3-D FIR FILTERS USING FFT AND WINDOW FUNCTIONS

3.1 INTRODUCTION

Discrete Fourier Transform (DFT) is a very important mathematical tool for the design and implementation of digital filters. Its importance arises because it can be efficiently computed by using the very powerful algorithm known as the Fast Fourier Transform (FFT) method[28][31]. DFT has numerous application in digital signal processing. However, our use of DFT will be in the design of digital filters, since the computation of the DFT require a considerable a mount of calculation FFT will be developed and used in the design. The method presented in this chapter extends the 2-D approach presented by M. A. Sid-Ahmed in [8]. The frequency response of linear Shift-Invariant (LSI)of 3-D system can be described b y

$$H(\omega_1,\omega_2,\omega_3) = \sum_{k_1=-\infty}^{k_1=\infty} \sum_{k_2=-\infty}^{k_2=\infty} \sum_{k_3=-\infty}^{k_3=\infty} h(k_1,k_2,k_3)\, e^{-j(\omega_1 k_1+\omega_2 k_2+\omega_3 k_3)} \tag{3.1}$$

For a causal system $h(k_1,k_2,k_3)$ exists only if $k_1 \geq 0, k_2 \geq 0$ and $k_3 \geq 0$ and is totally defined over a finite region of size N × N × N

$$H(\omega_1,\omega_2,\omega_3) = \sum_{k_1=0}^{k_1=N} \sum_{k_2=0}^{k_2=N} \sum_{k_3=0}^{k_3=N} h(k_1,k_2,k_3)\, e^{-j(\omega_1 k_1+\omega_2 k_2+\omega_3 k_3)} \tag{3.2}$$

That is $H(\omega_1,\omega_2,\omega_3)$ is totally defined over the region defined by $(-\pi \le \omega_1 \le \pi) \cap$ $(-\pi \le \omega_1 \le \pi) \cap (-\pi \le \omega_1 \le \pi)$ and is periodic in the frequency domain with a period of 2π along ω_1, ω_2 and ω_3.

$$\begin{aligned} H(\omega_1,\omega_2,\omega_3) &= H(\omega_1,\omega_2,\omega_3+2\pi) \\ &= H(\omega_1,\omega_2+2\pi,\omega_3) \\ &= H(\omega_1+2\pi,\omega_2,\omega_3) \\ &= H(\omega_1+2\pi,\omega_2+2\pi,\omega_3) \\ &= H(\omega_1,\omega_2+2\pi,\omega_3+2\pi) \\ &= H(\omega_1+2\pi,\omega_2,\omega_3+2\pi) \\ &= H(\omega_1+2\pi,\omega_2+2\pi,\omega_3+2\pi) \end{aligned}$$

If we sample uniformly in ω_1,ω_2 and ω_3 such that we have N × N × N samples in the defined region then we can write

$$\omega_1 = \frac{2\pi}{N}n_1 \quad , \omega_2 = \frac{2\pi}{N}n_2 \quad \text{and } \omega_3 = \frac{2\pi}{N}n_3 \tag{3.4}$$

Therefore

$$H(n_1,n_2,n_3) = \sum_{k_1=0}^{N-1}\sum_{k_2=0}^{N-1}\sum_{k_3=0}^{N-1} h(k_1,k_2,k_3)e^{-J\frac{2\pi}{N}(n_1k_1+n_2k_2+n_3k_3)} \tag{3.5}$$

which is the 3-D Discrete Fourier Transform (DFT) Or

$$H(n_1,n_2,n_3) = DFT\{h(k_1,k_2,k_3)\} \quad \text{for } n_i = 0,1...N-1 \quad i=1,2,3$$

The Inverse Discrete Fourier Transform (IDFT) is given by

$$h(k_1,k_2,k_3)=\frac{1}{N^3}\sum_{n_1=0}^{N}\sum_{n_2=0}^{N}\sum_{n_3=0}^{N}H(n_1,n_2,n_3)e^{j\frac{2\pi}{N}(k_1n_1+k_2n_2+k_3n_3)} \tag{3.6}$$

or $h(k_1,k_2,k_3)=IDFT\{H(n_1,n_2,n_3)\}$

3.2 Fast Fourier Transform (FFT)

The direct evaluation of the DFT involves N^2 complex multiplications and $(N-1)^2$ complex additions for each value of F(n). Consequently for large values of N direct evaluation will require a considerable amount of computations. The amount of computation is multiplied by 3 for 3-D DFT[48]The efficient approach for reducing the amount of computations in evaluations of DFT is through the use of efficient algorithm known as Fast Fourier Transform (FFT). Two algorithms are available for evaluating FFT mainly *decimation- in- time* and *decimation-in-frequency* .The two algorithms are described in[8] [16].

3.3 Three-Dimensional FFT

The procedure for obtaining 3-D FFT can be done as follows

for k_1=0 N-1 do the following

1- obtain 1-D FFT of each row and store the result in an intermediate array.

2- Transpose the intermediate array.

3- Obtain 1-D FFT of each row and store the result in an intermediate array.

4- Transpose the intermediate array.

End

The result is a 3-D FFT of the input signal.

From the definition of 3-D DFT in eq(3.5) and eq(3.6) it can be shown that

$$h(k_1,k_2,k_3)e^{j(2\pi/N)(ak_1+bk_2+ck_3)} \leftrightarrow H(n_1-a,n_2-b,n_3-c) \tag{3.7}$$

That is a linear phase shift in the frequency domain represents a constant shift in the space domain. From Eq. (3.7), consider the special case of $a=b=c=(N/2)$.

$$h(k_1,k_2,k_3)e^{j\pi(k_1+k_2+k_3)} = h(k_1,k_2,k_3)(e^{j\pi})^{(k_1+k_2+k_{33})} = h(k_1,k_2,k_3)(-1)^{(k_1+k_2+k_3)} \tag{3.8}$$

that is

$$h(k_1,k_2,k_3)(-1)^{(k_1+k_2+k_3)} \leftrightarrow H(n_1-\frac{N}{2},n_2-\frac{N}{2},n_3-\frac{N}{2}) \tag{3.9}$$

Eq. (3.9) indicates that by multiplying each element of the input array by $(-1)^{k_1+k_2+k_3}$ before taking the FFT, we will obtain a frequency spectrum that has its(0,0,0) frequency point in the center of the 3-D array. This property is very important in the design of FIR filters using Frequency Sampling Technique.

3.4 Design of 3-D FIR filters using 3-D FFT

A 3-D FIR filter can be designed using the following steps

1- Given the description of magnitude and phase spectrums, generate an M ×M × M array H(l,m,n) in which the zero frequency (0,0,0) is located at (M/2,M/2,M/2). M should be a multiple of 2.

2- Obtain the inverse FFT of $H(l,m,n)(-1)^{l+m+n}$. The result is the impulse response $h(l,m,n)(-1)^{l+m+n}$ centered about the point (M/2,M/2,M/2).

3- Given the order of the filter as N × N × N, N being an odd number, the filter's coefficients are the values of $h(l,m,n)$ located in the window extending from (M/2-(N-1)/2, M/2-(N-1)/2, M/2-(N-1)/2) to (M/2+(N-1)/2, M/2+(N-1)/2, M/2+(N-1)/2) The above procedure involves the use of what is known as *rectangular window* which is described by (3.10).

The FIR filter's coefficients are given by

$$hd=h(l,m,n).w(l,m,n)$$

$$w(l,m,n)=\begin{cases}1 & \text{for } \left[(\frac{M}{2}-\frac{N-1}{2})\le l\le(\frac{M}{2}+\frac{N-1}{2})\right]\cap \\ & \left[(\frac{M}{2}-\frac{N-1}{2})\le m\le(\frac{M}{2}+\frac{N-1}{2})\right]\cap \\ & \left[(\frac{M}{2}-\frac{N-1}{2})\le n\le(\frac{M}{2}+\frac{N-1}{2})\right] \\ 0 & \text{Otherwise}\end{cases} \qquad (3.10)$$

Design Example 3.1

We wish to design a 3-D FIR filter with order of 5 × 5 × 5 with the following specifications

$$H(\omega_1,\omega_2,\omega_3)=\begin{cases}0 & \text{for } 0\le\sqrt{\omega_1^2+\omega_2^2+\omega_3^2}\le 0.9 \\ 1 & \text{Otherwise}\end{cases}$$

Figure 3.1,3.2and 3.3 show 3-D plots of the magnitude response of the designed filter at different values of ω_3.

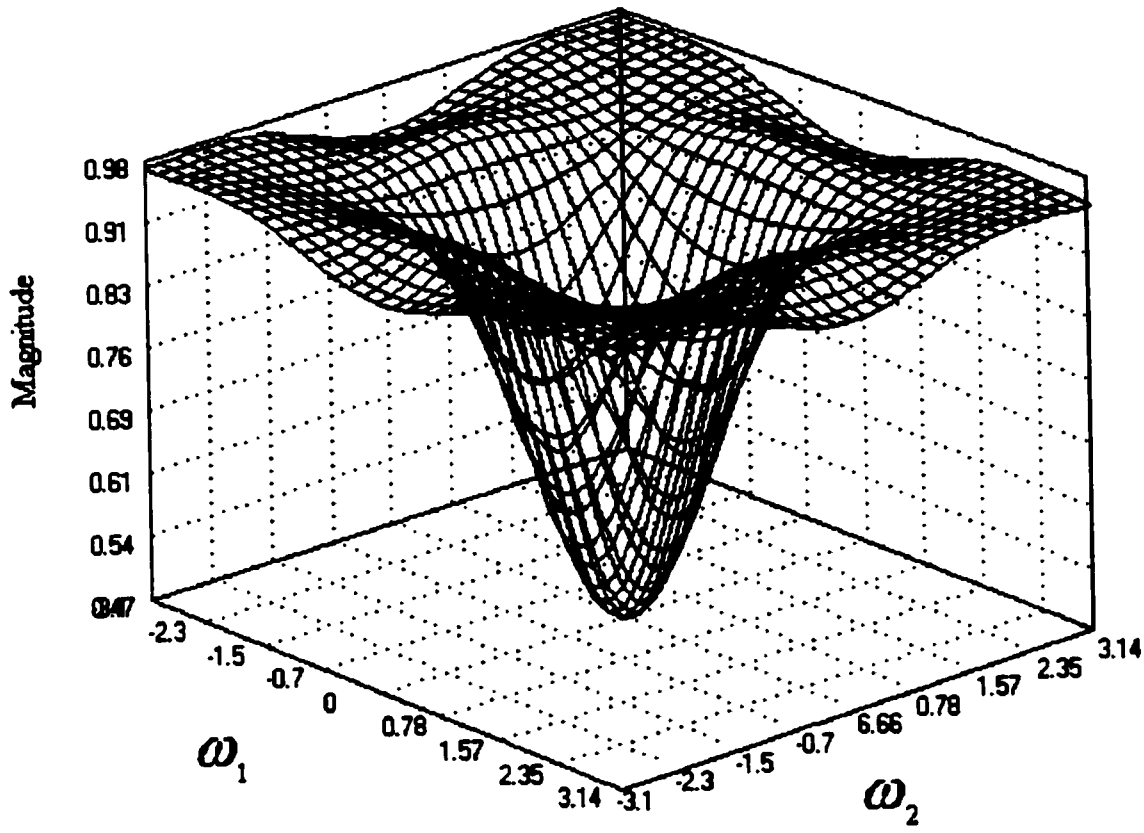

Figure 3.1 Magnitude response of 5 × 5 × 5 FIR filter with ω_c=0.9 at ω_3=0

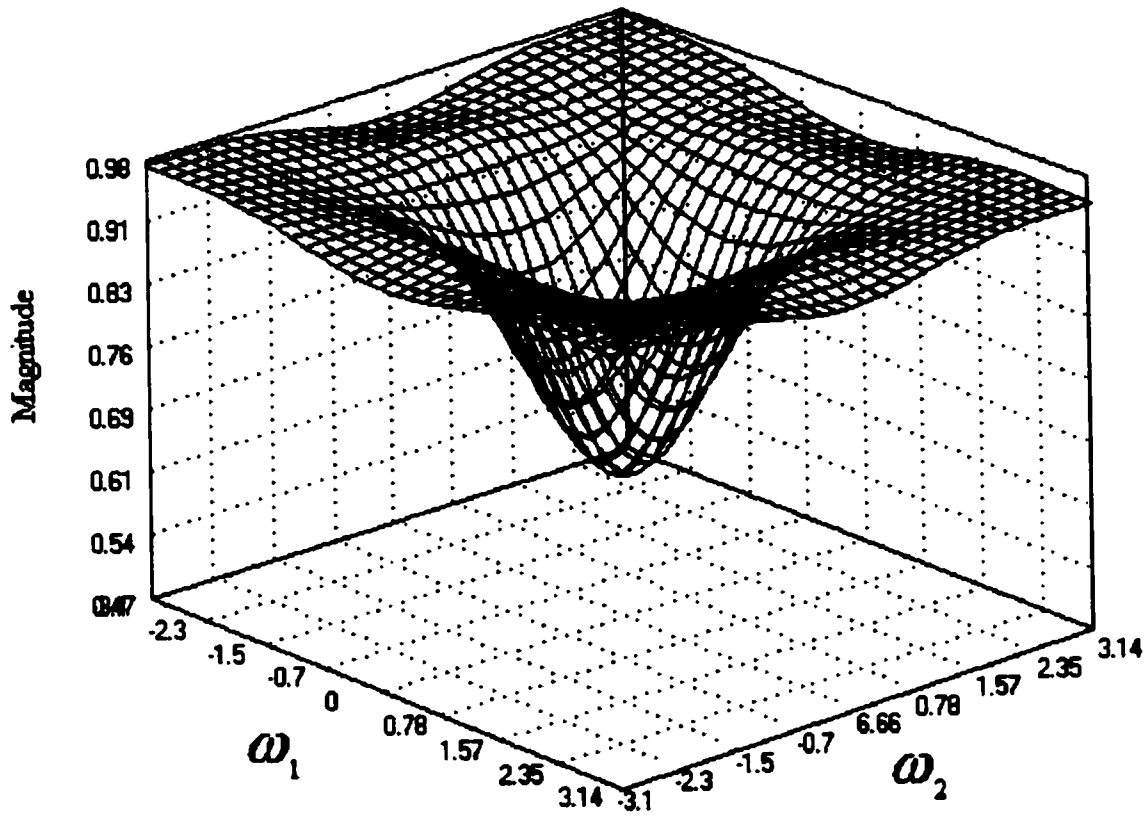

Figure 3.2 Magnitude response of 5 × 5 × 5 FIR filter with ω_c=0.9 at ω_3=0.7854

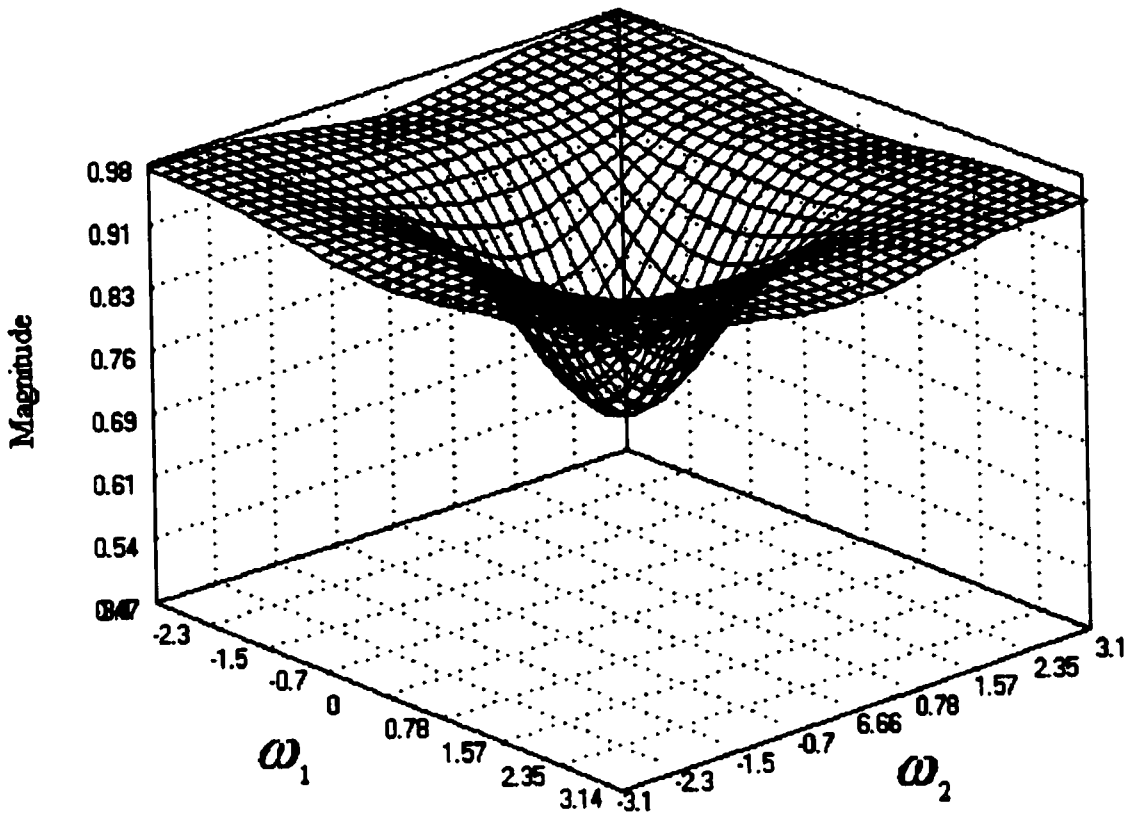

Figure 3.3 Magnitude response of 5 × 5 × 5 FIR filter with ω_c=0.9 at ω_3=0.982

The oscillations in the passband are quite noticeable. These oscillations are due to the slow convergence of Fourier coefficients and they are known as Gibbs oscillations. An easy-to-apply technique for smoothing the Gibb's oscillations is to apply a class of time-domain functions known as window functions.

3.5 Window Functions

Since the properties of 1-D and 3-D window functions are identical, a 1-D window functions can be used as a basis for generating 3-D window functions. There are two methods of obtaining 3-D window functions from 1-D window. The first method generates 3-D window functions by cascading three 1-d Windows as follows

$$w(l,m,n)=w_1(l).w_2(m).w_3(n) \tag{3.11}$$

The second method is by sampling a spherically rotated, continuos window function. Division by $\sqrt{3}$ is used to ensure that the magnitude of *l*, *m* or *n* does not exceed (N-1)/2 which is the maximum value for the 1-D case.

The most frequently used window function are

1-Rectangular Window.

2-Von hann & Hamming Window.

3- Blackman Window.

4-Kaiser Window.

3.5.1 Rectangular Window

$$w(l,m,n)=\begin{cases}1 & \text{for } (\frac{M}{2}-\frac{N-1}{2})\le l\le(\frac{M}{2}-\frac{N-1}{2}) \\ & (\frac{M}{2}-\frac{N-1}{2})\le m\le(\frac{M}{2}-\frac{N-1}{2}) \\ & (\frac{M}{2}-\frac{N-1}{2})\le n\le(\frac{M}{2}-\frac{N-1}{2}) \\ 0 & \text{elsewhere}\end{cases} \tag{3.12}$$

3.5.2 Hann and hamming windows

$$w(l,m,n)=\begin{cases}\alpha-(1-\alpha)\cos(\frac{2\pi\sqrt{l^2+m^2+n^2}}{(N-1)\sqrt{3}}) & |l|\le\frac{N-1}{2},|m|\le\frac{N-1}{2},|n|\le\frac{N-1}{2} \\ 0 & \text{Otherwise}\end{cases} \tag{3.13}$$

There are two different values for α in the Hann window α=0.5, and in Hamming window α=0.54. N is the order of the filter.

3.5.3 Blackmann window

Blackmann window is given by

$$w(l,m,n)=\begin{cases}0.42+0.5\cos\left(\frac{2\pi\sqrt{l^2+m^2+n^2}}{(N-1)\sqrt{3}}\right)+0.08\cos\left(\frac{4\pi\sqrt{l^2+m^2+n^2}}{(N-1)\sqrt{3}}\right) & |l|\le\frac{N-1}{2},|m|\le\frac{N-1}{2},|n|\le\frac{N-1}{2}\\ 0 & \text{Otherwise}\end{cases} \quad (3.14)$$

The additional cosine term is used to increase the reduction of the amplitude of the Gibbs oscillation

3.5.4 Kaiser window

The kaiser window is given by

$$w(l,m,n)\begin{cases}\frac{I_0(\beta)}{I_0(\alpha)} & |l|\le\frac{N-1}{2},|m|\le\frac{N-1}{2}\text{ and }|n|\le\frac{N-1}{2}\\ 0 & \text{otherwise}\end{cases} \quad (3.15)$$

α is an independent parameter and β is given by

$$\beta=\alpha\sqrt{1-\left(\frac{2\sqrt{l^2+m^2+n^2}}{(N-1)\sqrt{3}}\right)^2}$$

$I_0(x)$ is the zeroth order Bessel function of the first kind and can be evaluated by

$$I_0(x)=1+\sum_{k=1}^{\infty}\left[\frac{1}{k!}\left(\frac{x}{2}\right)^k\right]^2$$

Design Example 3.2

We wish to design a 3-D FIR filter with order of $7 \times 7 \times 7$ with the following specifications

$$|H(e^{j\omega_1}, e^{j\omega_2}, e^{j\omega_3})| = \begin{cases} 1 & \text{for } 0 \le \sqrt{\omega_1^2 + \omega_2^2 + \omega_3^2} \le 1 \\ 0 & \text{Otherwise} \end{cases}$$

Using Blackmann's window the resultant magnitude response is shown in figures 3.4-3.7 at different values of ω_3 .

3.6 SUMMARY

Extension of 1-D FFT to 3-D was done. Three-Dimensional FFT was utilized in the design of 3-D FIR filters. The designed filters exhibited high ripples in pass-band and stop-band. which known as *Gibbs oscillations.* To reduce the Gibbs oscillations,3-D window functions were used. From figures 3.4-3.7 we can see that the reduction in Gibbs oscillations is noticeable.

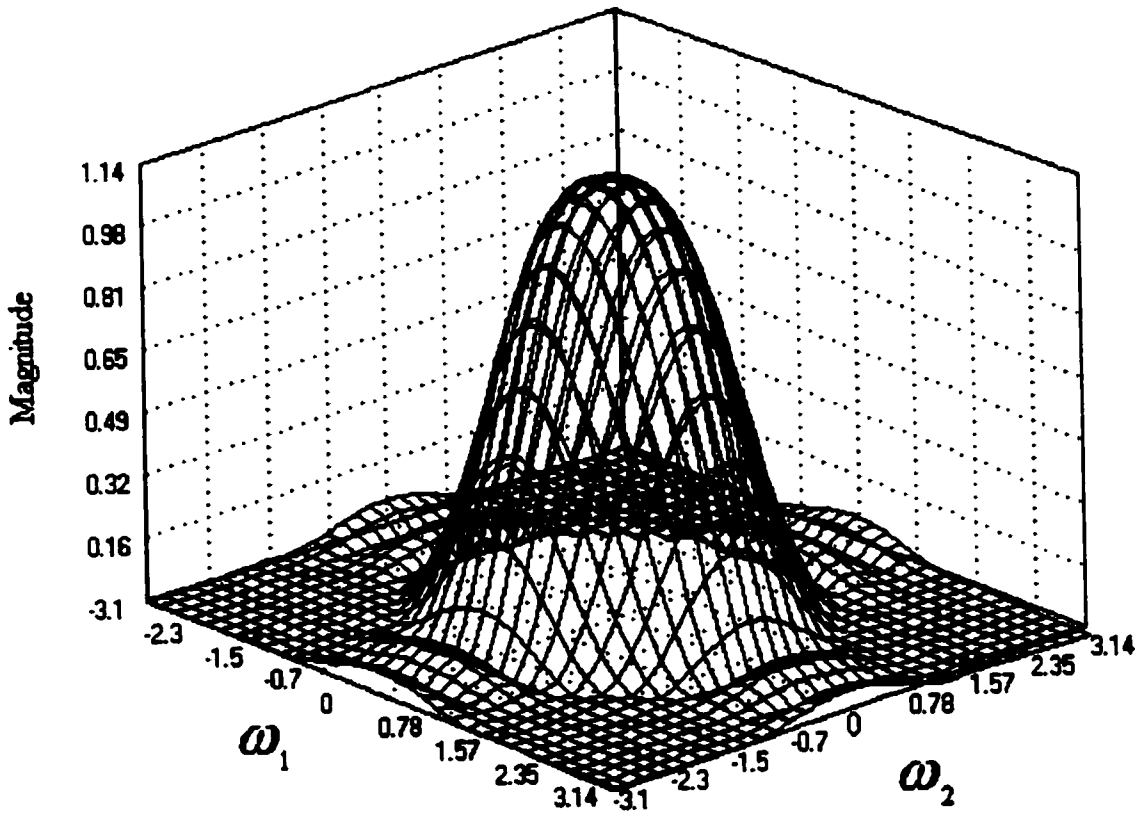

Figure 3.4 Magnitude response of $7 \times 7 \times 7$ FIR filter with ω_c=1.4 at ω_3=0 designed using Blackman window

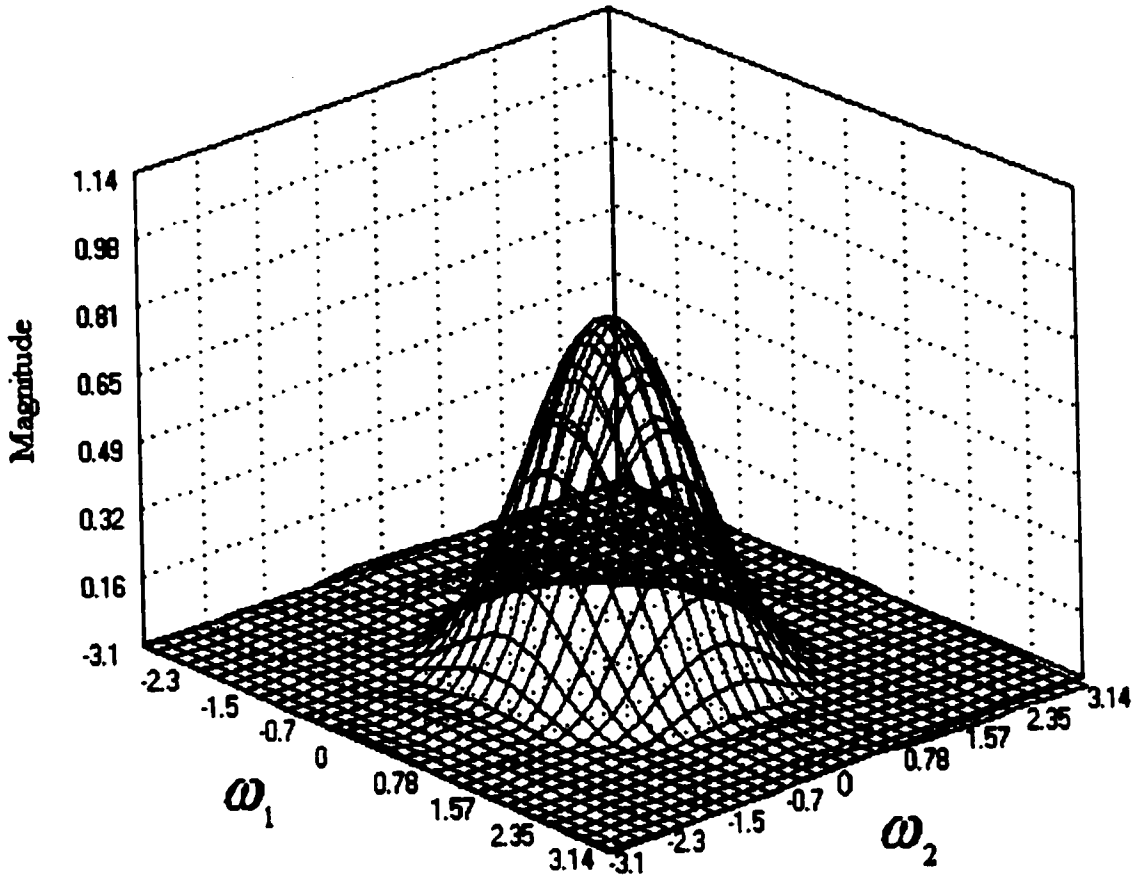

Figure 3.5 Magnitude response of $7 \times 7 \times 7$ FIR filter with ω_c=1.4 at ω_3=0.98

designed using Blackman window

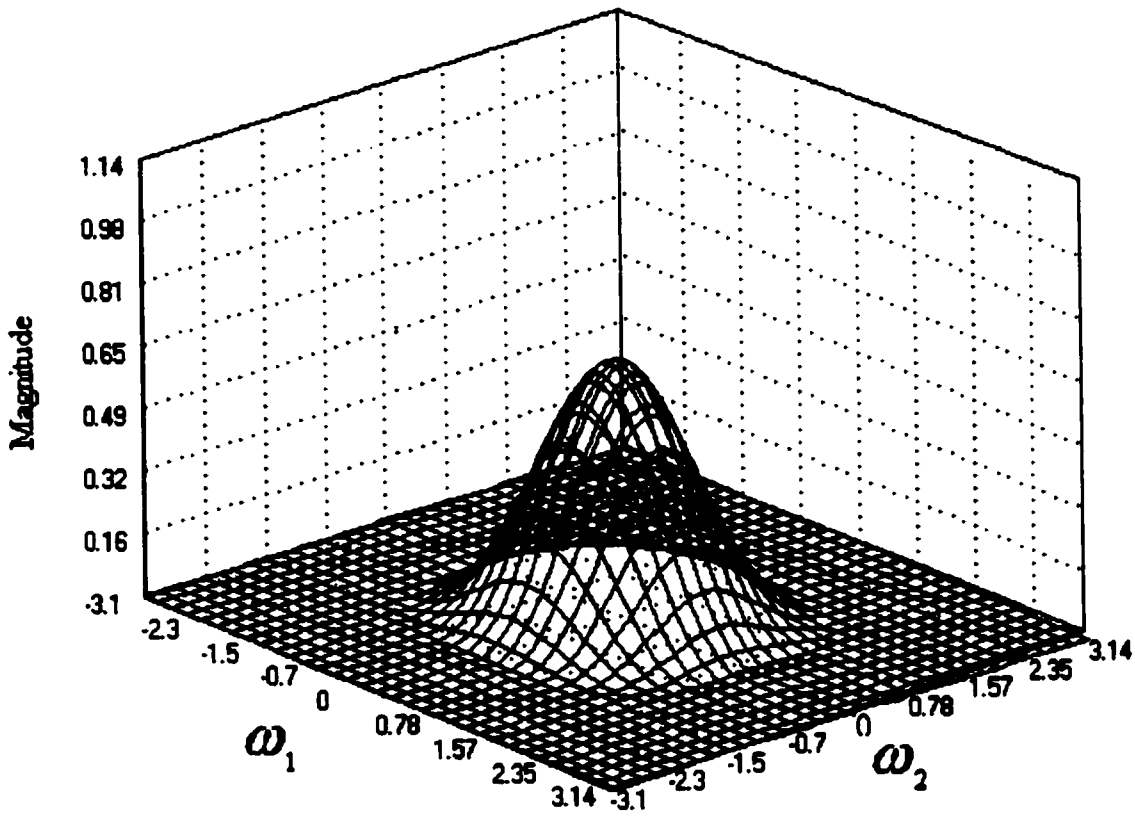

Figure 3.6 Magnitude response of 7 × 7 × 7 FIR filter with ω_c=1.4 at ω_3=1.1781

designed using Blackman window

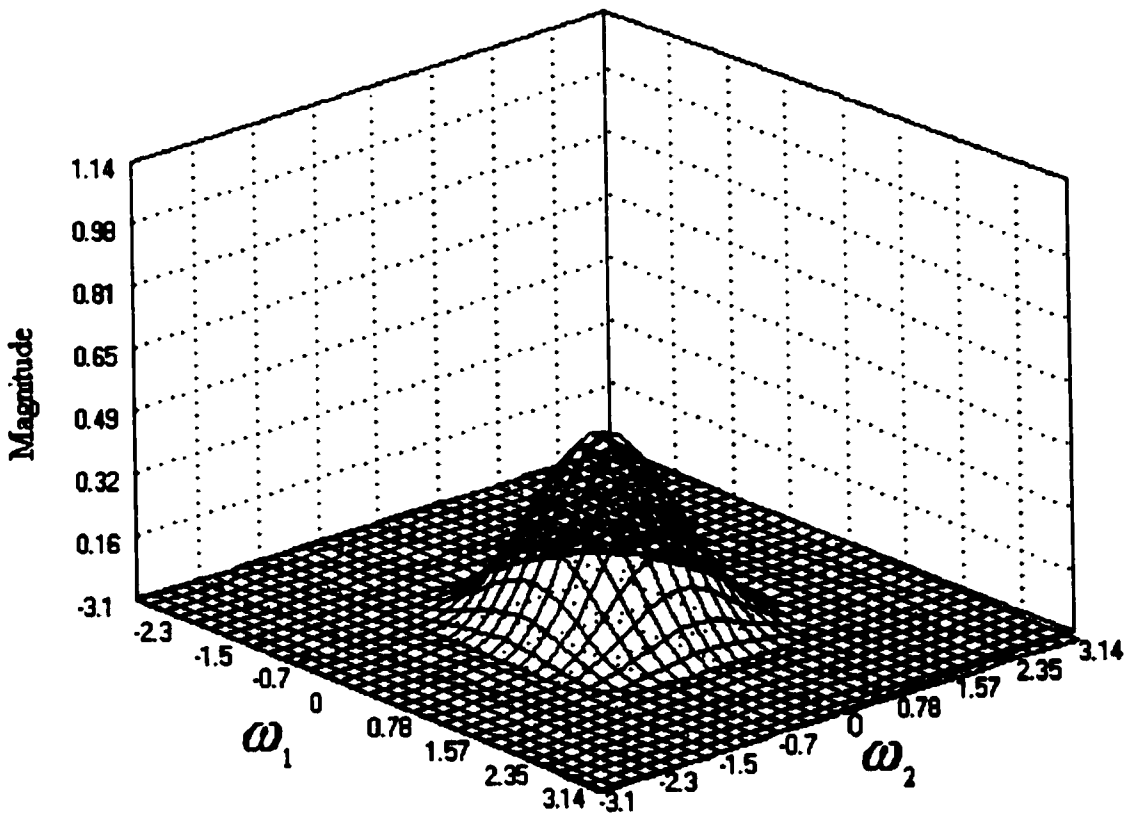

Figure 3.7 Magnitude response of 7 × 7 × 7 FIR filter with ω_c=1.4 at ω_3=1.3745

designed using Blackman window

CHAPTER FOUR

DESIGN OF 3-D IIR FILTERS USING SHANK'S METHOD

4.1 INTRODUCTION

Stable and linear phase FIR filers were designed in the previous chapter using FFT. As can be seen low-order FIR filters exhibit a wide transition bands and ripples in the pass band which are considered as disadvantages in image processing. Due to the slow convergence of Fourier series, truncating the series will result in Gibbs oscillation.

Although window functions are used to smoothen these oscillations, steep transition band can only be obtained using higher order FIR filters. Therefore, the only way to obtain a steep transition FIR is to use high order which translates to high cost.

An alternative way to avoid high cost filters is to use infinite impulse response transfer function or (IIR) filters. Such filters can provide steeper transition bands with lower order than equivalent FIR filters . Although FIR filters have the disadvantages of high cost in term of hardware realizations, they have the advantage of stability and linear phase. Since IIR filters have infinite impulse response, they are prone to instability. That is, the impulse response could grow indefinitely if care is not taken in their design. Linear phase which is easy to obtain in FIR filters is hard to come by in IIR filters. The low cost and fast filtering make the IIR filters attractive. In this chapter we will study the design of stable IIR with near linear phase characteristic by extending the approach presented for 2-D in[8] to 3-D.

4.2 Three -Dimensional IIR Filter

The transfer function of 3-D IIR filters is given by

$$H(z_1,z_2,z_3)=\frac{\sum_{i=0}^{N}\sum_{j=0}^{N}\sum_{k=0}^{N}a(i,j,k)z_1^{-1}z_2^{-1}z_3^{-1}}{\sum_{i=0}^{N}\sum_{j=0}^{N}\sum_{k=0}^{N}b(i,j,k)z_1^{-1}z_2^{-1}z_3^{-1}} \tag{4.1}$$

With out any loss of generality b(0,0,0) can be set to 1.

If $x(l,m,n)$ and $y(l,m,n)$ are the input and output sequences, respectively, then from Eq(4.1) we can write

$$y(l,m,n)=\sum_{i=0}^{N}\sum_{j=0}^{N}\sum_{k=0}^{N}a(i,j,k)x(l-i,m-j,n-k)-\sum_{\substack{i=0\\ i+j+k\neq 0}}^{N}\sum_{j=0}^{N}\sum_{k=0}^{N}b(i,j,k)x(l-i,m-j,n-k) \tag{4.2}$$

For a filter to be stable its impulse response $h(l,m,n)$ must have the following characteristics

1- It mus be causal

$$h(l,m,n)=0 \quad (l<0)\cup(m<0)\cup(n<0) \tag{4.3}$$

2-It must be absolutely summable.

$$\sum_{l=0}^{N}\sum_{m=0}^{N}\sum_{n=0}^{N}|h(l,m,n)| \quad < \quad \infty \tag{4.4}$$

If $h(l,m,n)$ decays with the increase of l, m, and n there is a chance that the filter designed to approximate it will be a stable filter and the condition given by Eq. (4.4) will be satisfied

Next we need to impose some specification of the phase $\phi(\omega_1,\omega_2,\omega_3)$ to end up with a stable filter with near linear phase or a constant group delay, where the group delay is defined as

$$\tau_1(\omega_1,\omega_2,\omega_3) = \frac{\partial\phi(\omega_1,\omega_2,\omega_3)}{\partial\omega_1}$$

$$\tau_2(\omega_1,\omega_2,\omega_3) = \frac{\partial\phi(\omega_1,\omega_2,\omega_3)}{\partial\omega_2} \qquad (4.5)$$

$$\tau_3(\omega_1,\omega_2,\omega_3) = \frac{\partial\phi(\omega_1,\omega_2,\omega_3)}{\partial\omega_3}$$

Selecting constant values for the group delay, near linear phase is obtained. we now investigate a well-known method for filter design and its modifications as described by Sid-Ahmed in[8] in order to include a near linear phase

4.3 Shank's Method

The IIR filter described by Eq.(4.1) can be obtained by letting $z_1 = e^{-j\omega_1\pi}$, $z_2 = e^{-j\omega_2\pi}$ and $z_3 = e^{-j\omega_3\pi}$. Hence

$$H(\omega_1,\omega_2,\omega_3) = \frac{A(\omega_1,\omega_2,\omega_3)}{B(\omega_1,\omega_2,\omega_3)} \qquad (4.6)$$

Where

$$A(\omega_1,\omega_2,\omega_3)=\sum_{i_1=0}^{N}\sum_{i_2=0}^{N}\sum_{i_3=0}^{N}a(i_1,i_2,i_3)e^{-j(\omega_1 i_1\pi)}e^{-j(\omega_2 i_2\pi)}e^{-j(\omega_3 i_3\pi)} \tag{4.7}$$

and

$$B(\omega_1,\omega_2,\omega_3)=\sum_{i_1=0}^{N}\sum_{i_2=0}^{N}\sum_{i_3=0}^{N}b(i_1,i_2,i_3)e^{-j(\omega_1 i_1\pi)}e^{-j(\omega_2 i_2\pi)}e^{-j(\omega_3 i_3\pi)} \tag{4.8}$$

If we let H^d represent the desired frequency, the error between the desired frequency and the designed frequency can be defined as

$$\varepsilon(\omega_1,\omega_2,\omega_3)=H^d(\omega_1,\omega_2,\omega_3)-\frac{A(\omega_1,\omega_2,\omega_3)}{B(\omega_1,\omega_2,\omega_3)} \tag{4.9}$$

Translating Eq. (4.9) in the space domain yields

$$\varepsilon(l,m,n)=h^d(l,m,n)-h(l,m,n) \tag{4.10}$$

where *h(l,m,n)* is the impulse response of the filter and can be generated from

$$h(l,m,n)=\sum_{i=0}^{N}\sum_{j=0}^{N}\sum_{k=0}^{N}a(i,j,k)\delta(l-i,m-j,n-k)-\sum_{\substack{i=0\\ i+j+k=0}}^{N}\sum_{j=0}^{N}\sum_{k=0}^{N}b(i,j,k)\delta(l-i,m-j,n-k) \tag{4.11}$$

Forming the L_2 norm

$$Q=\sum_{l=0}^{M}\sum_{m=0}^{M}\sum_{n=0}^{M}\varepsilon^2(l,m,n) \tag{4.12}$$

Where M×M×M is the number of samples provided in the impulse response .

To obtain the coefficients of the filter Q must be minimized. This is done by differentiating Q with respect to each filter coefficients and equating the result to zero as follows

$$\frac{dQ}{da_{t,u,v}} = 2\sum_{l=0}^{M-1}\sum_{m=0}^{M-1}\sum_{n=0}^{M-1}\varepsilon(l,m,n)\frac{d\varepsilon(l,m,n)}{da_{t,u,v}} \qquad (4.13)$$

$$t,u,v = 0,1\ldots N$$

$$\frac{dQ}{db_{t,u,v}} = 2\sum_{l=0}^{M-1}\sum_{m=0}^{M-1}\sum_{n=0}^{M-1}\varepsilon(l,m,n)\frac{d\varepsilon(l,m,n)}{db_{t,u,v}} \qquad (4.14)$$

$$t,u,v = 0,1\ldots N \qquad t+v+u \neq 0$$

Equation (4.13)will generate (N+1)×(N+1)×(N+1) non-linear equations in $\{a_{ijk}\}$ and equation(4.14) will generate (N+1)×(N+1)×(N+1)-1 non-linear equations in $\{b_{ijk}\}$. Since we have (N+1)×(N+1)×(N+1) $\{a\}$ coefficients and (N+1)×(N+1)×(N+1)-1 $\{b\}$ coefficients, solving the system of equations will yield the required filter coefficients.

To avoid the highly non-linear equations Shank proposed minimization of a weighted error rather than the minimization of the true error. This is done by multiplying Eq(4.9) by $B(\omega_1,\omega_2,\omega_3)$ to obtain

$$\varepsilon(\omega_1,\omega_2,\omega_3)B(\omega_1,\omega_2,\omega_3) = B(\omega_1,\omega_2,\omega_3)H^d(\omega_1,\omega_2,\omega_3) - A(\omega_1,\omega_2,\omega_3) \qquad (4.15)$$

Replacing the weighted error $\varepsilon(\omega_1,\omega_2,\omega_3)B(\omega_1,\omega_2,\omega_3)$ by $\varepsilon'(\omega_1,\omega_2,\omega_3)$ and translating the result into space domain Eq.(4.15) can be written as

$$\varepsilon'(l,m,n)=\sum_{i=0}^{N}\sum_{j=0}^{N}\sum_{k=0}^{N}b_{ijk}h^d(l-i,m-j,n-k)-\sum_{i=0}^{N}\sum_{j=0}^{N}\sum_{k=0}^{N}a_{ijk}\delta(l-i,m-j,n-k) \quad (4.16)$$

forming the L_2 norm

$$Q=\sum_{l=0}^{M-1}\sum_{m=0}^{M-1}\sum_{n=0}^{M-1}\varepsilon'^{\,2}(l,m,n) \quad (4.17)$$

To minimize Q we differentiate with respect to filter coefficients and equate the result to zero

$$\frac{\partial Q}{\partial a_{tuv}}=2\sum_{l=0}^{M-1}\sum_{m=0}^{M-1}\sum_{n=0}^{M-1}\left[\sum_{i=0}^{N}\sum_{j=0}^{N}\sum_{k=0}^{N}a_{ijk}\delta(l-i,m-j,n-k)-\sum_{i=0}^{N}\sum_{j=0}^{N}\sum_{k=0}^{N}b_{ijk}h^d(l-i,m-j,n-k)\right]\delta(l-i,m-j,n-k) \quad (4.18)$$

for $t,u,v = 0,1,.....N$

Eq.(4.18) can be written as

$$a_{tuv}=\sum_{\substack{i=0\\ l,m,n\in R}}^{N}\sum_{j=0}^{N}\sum_{k=0}^{N}b_{ijk}h^d(l-i,m-j,n-k)$$

R is the unshaded area shown in fig(4.1).

Eq (4.17) can be written as

$$Q = \sum_{l=0}^{M-1}\sum_{m=0}^{M-1}\sum_{n=0}^{M-1}\left[\sum_{i=0}^{N}\sum_{j=0}^{N}\sum_{k=0}^{N} b_{ijk}\, h^{d}(l-i, m-j, n-k)\right] \qquad (4.19)$$
$$n \notin R$$

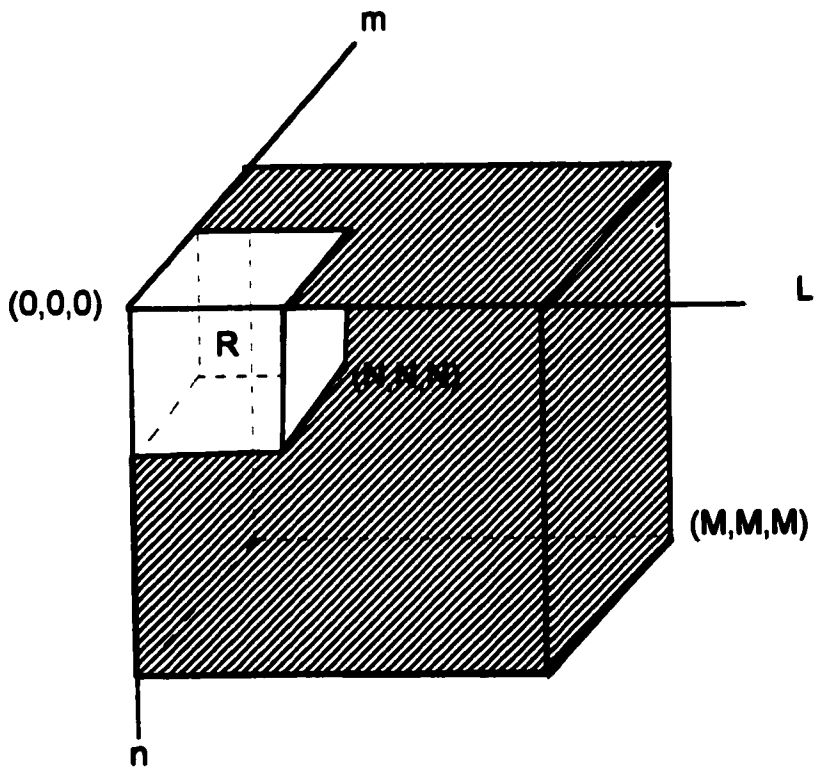

Figure 4.1 Region R for Eq. 4.19

To minimize Q we differentiate with respect to filter coefficients ant equate the result to zero as follows

$$\frac{\partial Q}{db_{tuv}} = 2\sum_{\substack{l \\ l,m,n \in R}} \sum_{m} \sum_{n} \left[\sum_{i=0}^{N} \sum_{j=0}^{N} \sum_{k=0}^{N} b_{ijk} h^d(l-i,m-j,n-k) \right] h^d(l-t,m-u,n-v) \quad (4.21)$$

Which reduces to

$$\sum_{i=0}^{N} \sum_{j=0}^{N} \sum_{k=0}^{N} b_{i,j,k} \sum_{l} \sum_{m} \sum_{n} h^d(l-i,m-j,n-k) h^d(t-l,u-m,v-n) = \sum_{l} \sum_{m} \sum_{n} h^d(l,m,n) h^d(l-i,m-j,n-k)$$
$$t,u,v = 0,1...N,\ t+u+v \neq 0 \quad (4.22)$$

Equation (4.22) generates (N+1)×(N+1)×(N+1)-1 linear equation which can be solved to produce the denominator coefficients . Numerator coefficients are obtained for Eq.(4.19)

4.4 Generating the impulse response [8]

The algorithm used for generating the impulse response is similar to that use for calculating the FIR filter coefficients. The procedure is as follows:

1- Generate M×M×M for the desired magnitude response $H^d(l,m,n)$ with (0,0,0) frequency points at (M/2,M/2,M/2), M should be multiple of 2.

2-Obtain the IFFT of $H^d(l,m,n)(-1)^{(l+m+n)}$, by assuming zero-phase spectrum.

3-Multiply the result by $(-1)^{(l+m+n)}$. This will provide us with the impulse response having its origin at (M/2,M/2,M/2).

Since the magnitude spectrum is considered periodic by the IFFT, the impulse response is equal to the Fourier series of a periodic function which will always converge and approach zero at least as fast as $\frac{c}{\sqrt{l^2+m^2+n^2}}$ where c is a constant independent of l, m and n. This means that the impulse response $h^d(l,m,n)$, is decreasing radially from the center of the array. Shifting the impulse response in space domain is equivalent the addition of a linear phase in the frequency domain. To verify this property we will apply the above approach to a low-pass filter characterized by a Butterworth function

$$H(\omega_1,\omega_2,\omega_3) = \frac{1}{1+0.414\left[\frac{\sqrt{\omega_1^2+\omega_2^2+\omega_3^2}}{\omega_c}\right]^{2n}} \qquad (4.25)$$

Using n=2 and ω_c=1.0 rad/unite the space domain impulse response is shown in figure 4.2, at mid-point of N.

Design Example 4.1

Design a 3-D IIR filter using the modified Shank's method with the following specifications

$$H(\omega_1,\omega_2,\omega_3)=\begin{cases}1 & \text{for } 0\le\sqrt{\omega_1^2+\omega_2^2+\omega_3^2}\le 1.4\\ 0 & \text{otherwise}\end{cases}$$

Using the outlined approach and using 2x2x2 filter order the coefficient in table 4.1 were obtained. Fig.4.3-4.5 show the magnitude response of the designed filter at different values of ω_3 and fig 3.6 show the phase response of the designed filter.

4.5 SUMMARY

In chapter we extended the modified Shank's method to 3-D. By assigning a constant shift in the impulse response in the space domain, near linear phase was possible in the frequency domain. It is also obvious from the figures that steeper transition band could be obtained using lower order IIR filter as compared to FIR filters of the same order.

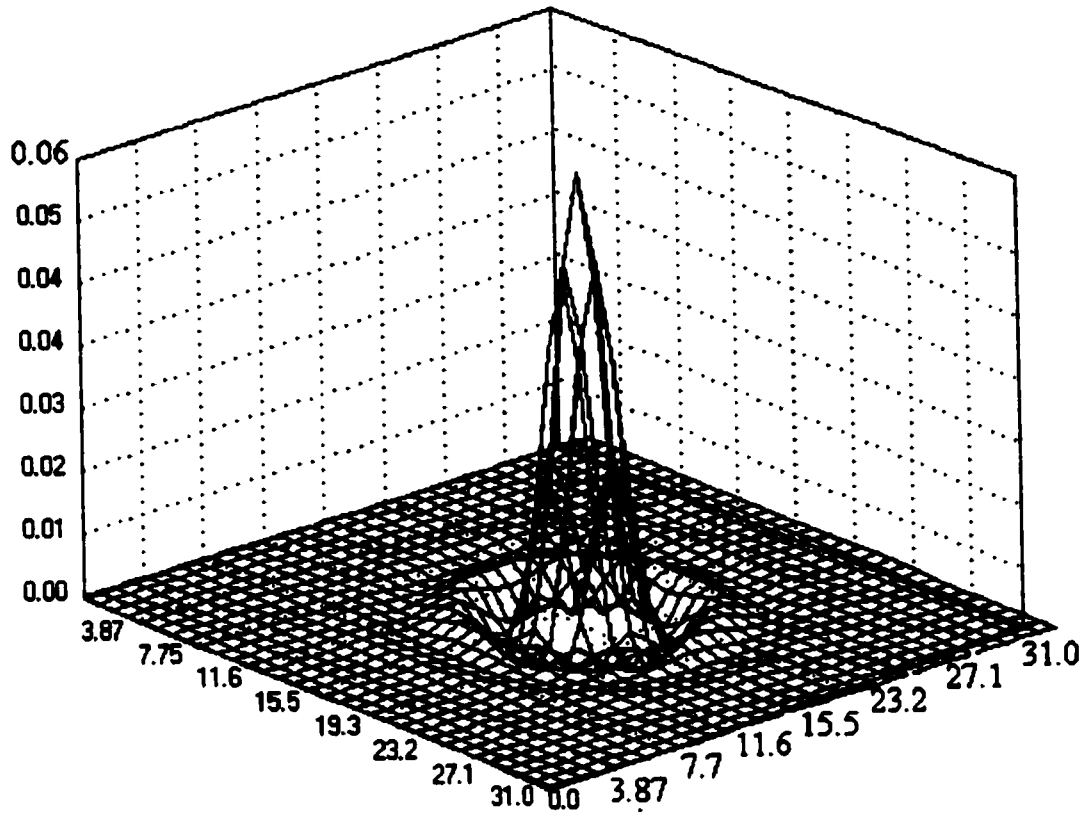

Figure 4.2 impulse response of the filter at N=16

Table 4.1 The coefficients of the filter in example 4.1

Index	A Coefficients	B Coefficients	Index	A Coefficients	B Coefficients
(0,0,0)	0.02572	1.0	(2,1,2)	0.0020834	-0.1328
(0,0,1)	0.01367	-0.8077	(2,2,0)	0.002088	0.1466
(0,0,2)	0.00643	0.33185	(2,2,1)	0.0020834	-0.1329
(0,1,0)	0.01367	-0.8077	(2,2,2)	0.000667	0.0702
(0,1,1)	0.00737	0.6973			
(0,1,2)	0.00514	-0.29415			
(0,2,0)	0.006432	0.33185			
(0,2,1)	0.005144	-0.29415			
(0,2,2)	0.002087	0.14657			
(1,0,0)	0.01367	-0.8077			
(1,0,1)	0.00738	0.6973			
(1,0,2)	0.00514	-0.29416			
(1,1,0)	0.007376	0.69727			
(1,1,1)	0.00423	-0.6433			
(1,1,2)	0.00304	0.27505			
(1,2,0)	0.0051	-0.29415			
(1,2,1)	0.00304	0.27504			
(1,2,2)	0.0020	-0.1328			
(2,0,0)	0.0064	0.3318			
(2,0,1)	0.0051	-0.29415			
(2,0,2)	0.002087	0.1466			
(2,1,0)	0.00514	-0.29415			
(2,1,1)	0.00304	0.27505			

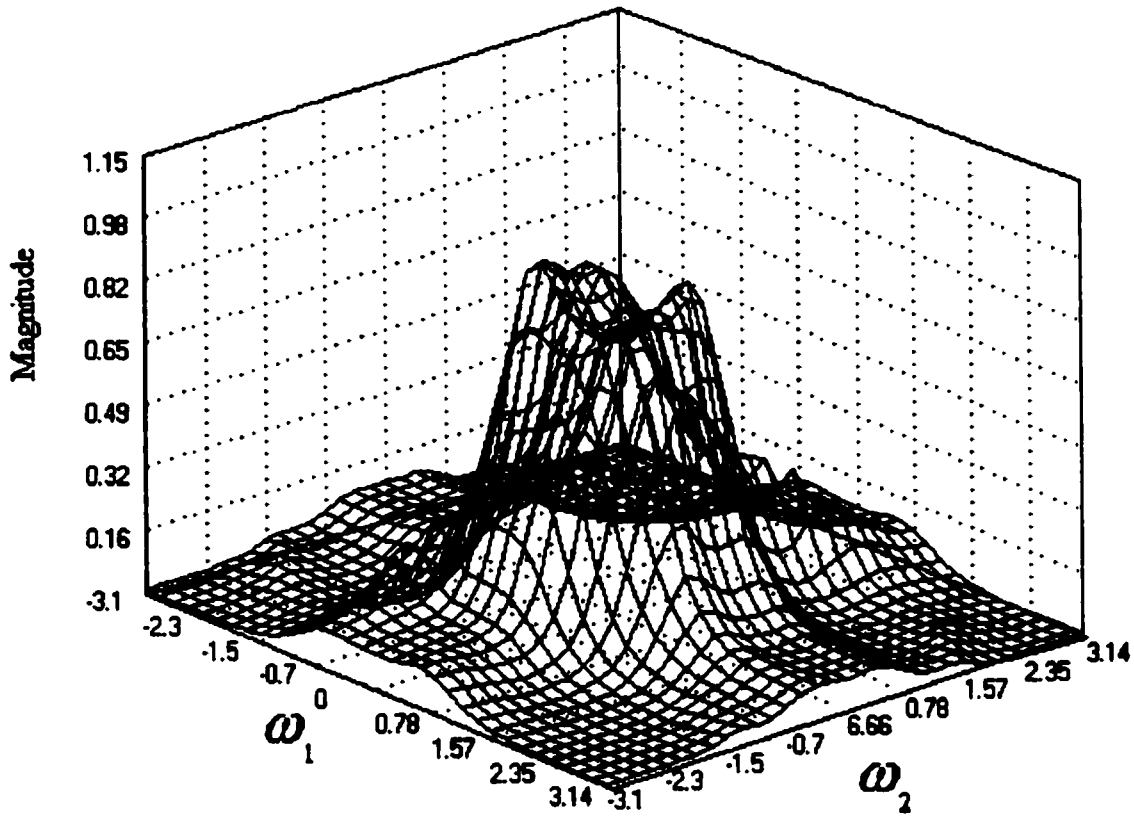

Figure 4.3 Magnitude response of 2×2×2 IIR filter with ω_c =1.4 at ω_3 =0

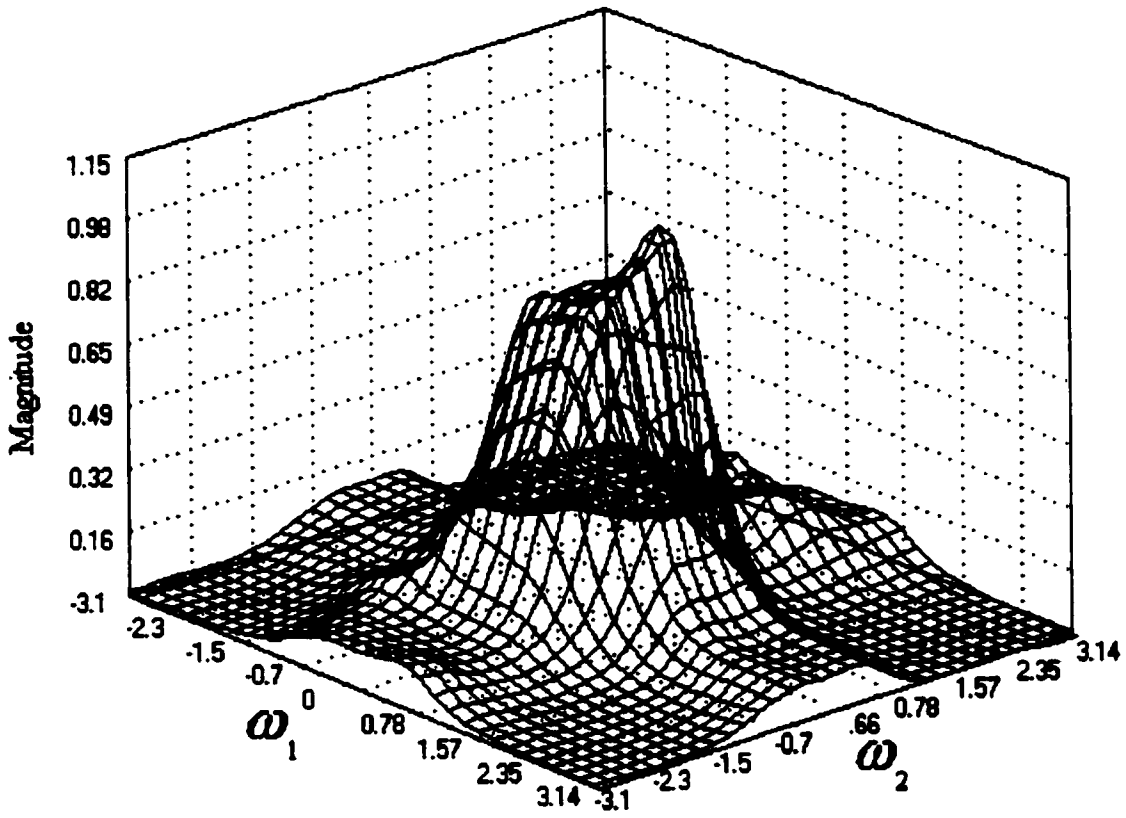

Figure 4.4 Magnitude response of 2×2×2 IIR filter with ω_c =1.4 at ω_3 =0.7854

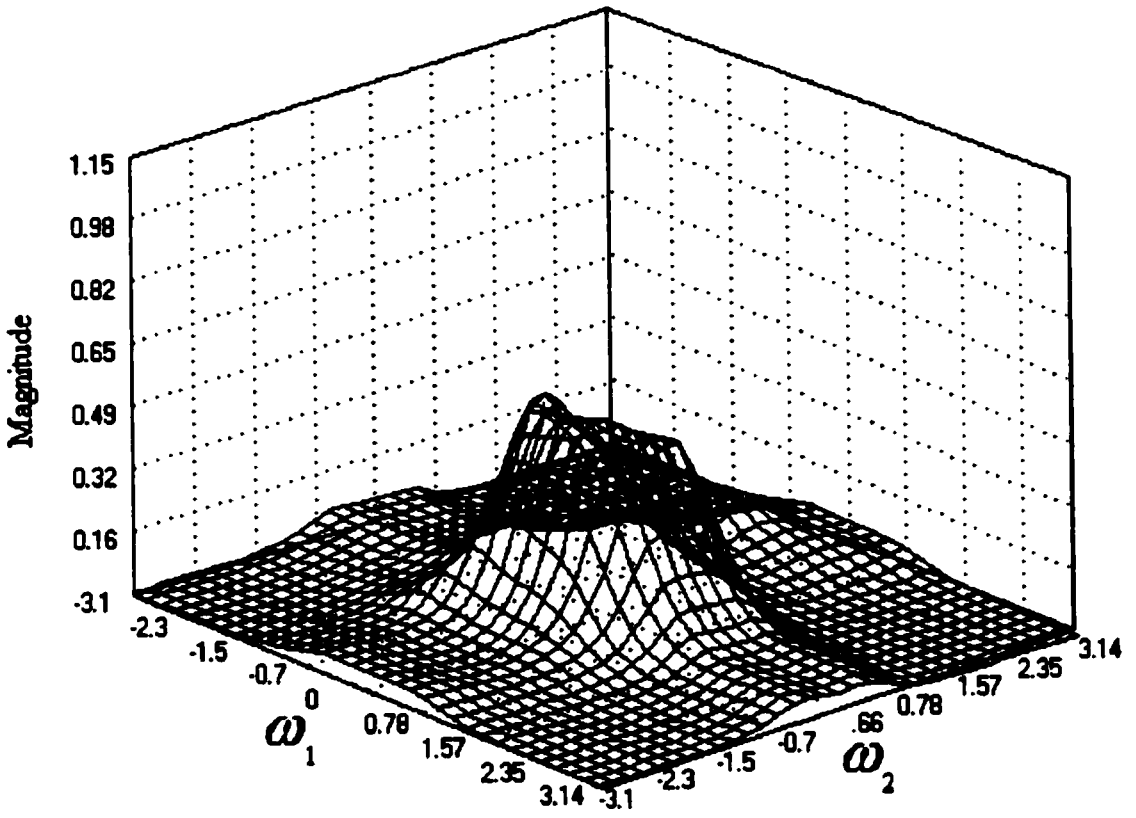

Figure 4.5 Magnitude response of 2×2×2 IIR filter with ω_c =1.4 at ω_3 =1.1781

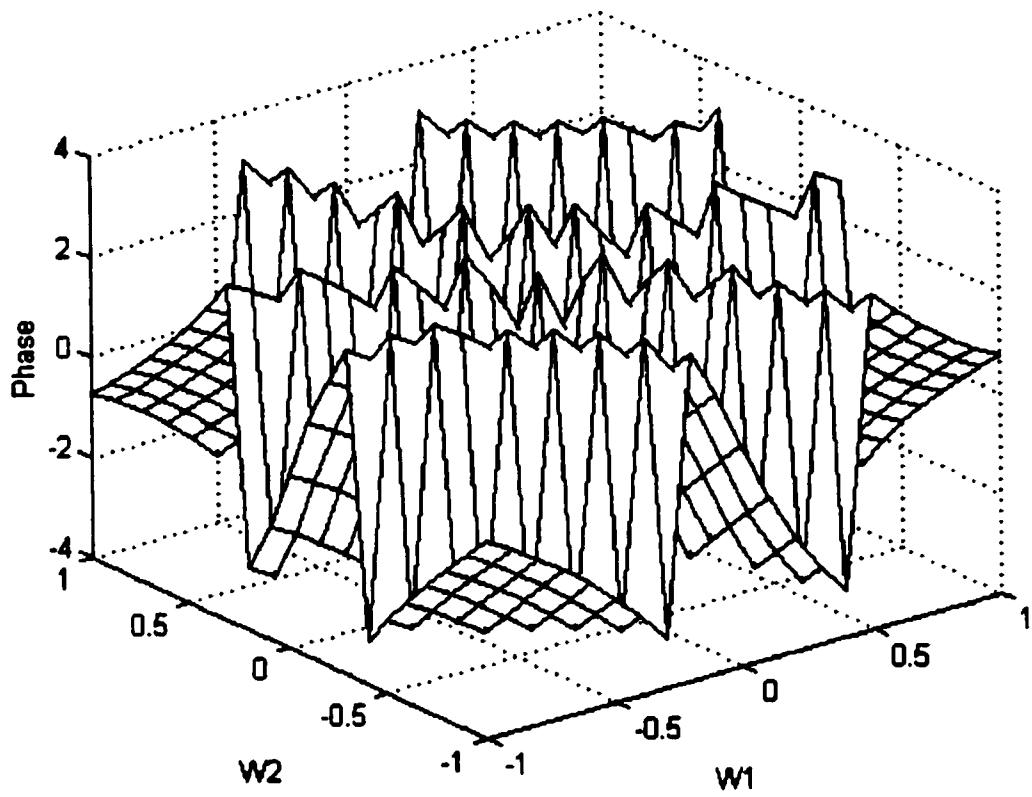

Figure 4.6 Phase response of the filter designed in example 4.1 at ω_3 =0

CHAPTER FIVE

DESIGN OF 3-D FILTERS USING LINEAR PROGRAMMING APPROACH

5.1 INTRODUCTION

Linear programming has been used in digital filter design[37,38,39]. In this chapter, a frequency domain design method utilizing a linear programming approach for the design of 3-D FIR filter with linear phase is presented. A simultaneous approximation of linear phase and arbitrary magnitude response is performed by minimizing an objective function which is a measure of the difference between the desired response and the approximated response. As the value of the objective function approach zero, the resulting magnitude and phase response approach the desired response. These methods are iterative in nature and, as a result, they usually require a considerable amount of computations. However, the use of symmetries will reduce the amount of computation drastically. The application of using programming in 3-D filter design is suitable for the filters having arbitrary amplitude and phase responses.

This chapter begins with a brief introduction to linear programming with the formulation of the design problem into linear programming problem and proceeds with fairly detailed description of the proposed algorithms. A commercially available package named LINDO will be used in the design examples.

5.2 Linear Programming

A linear programming problem can be mathematically stated as follows[49]

Minimize the objective function

$$c_1 x_1 + c_2 x_2 + c_3 x_3 + \ldots + c_j x_j \ldots\ldots + c_n x_n \qquad (5.1a)$$

subject to

$$\begin{aligned} a_{11} x_1 + a_{12} x_2 + \ldots + a_{1j} x_j + \ldots + a_{1n} x_n &= b_1 \\ a_{21} x_1 + a_{22} x_2 + \ldots + a_{2j} x_j + \ldots + a_{2n} x_n &= b_2 \\ a_{31} x_1 + a_{32} x_2 + \ldots + a_{3j} x_j + \ldots + a_{3n} x_n &= b_3 \\ \ddot{a}_{m1} x_1 + a_{m2} x_2 + \ldots + a_{mj} x_j + \ldots + a_{mn} x_n &= b_m \end{aligned} \qquad (5.1b)$$

and

$$x_1, x_2, x_3 \ldots\ldots x_n \geq 0 \qquad (5.1c)$$

where

c_j, b_j and a_{ij} are constants

alternatively linear programming problem may be stated in vector form as follows

Minimize the objective function

$$CX$$

subject to

$$AX=B$$
$$X \geq 0$$

The main advantage of linear programming is that the solution to the formulation is guaranteed to be global optimum. In this chapter we shall utilize linear programming approach in the design of 3-D FIR filters and IIR filters.

5.3 Design of 3-D FIR Filters

5.3.1 Linear programming formulation

we now formulate the filter design problem in such a way that it can be solved by linear programming. An optimum set of coefficients that gives the best approximation of $M_I(\omega_1,\omega_2,\omega_3)$ to the desired magnitude function $M_D(\omega_1,\omega_2,\omega_3)$ will be the solution to our problem.

If the filter transfer function is given in the form of equation (1.3), then the magnitude-squared response of the filter can be obtained through the following steps:

$$M_D^2(\omega_1,\omega_2,\omega_3)=H_D(z_1,z_2,z_3)H_D(z_1^{-1},z_2^{-1},z_3^{-1})\Big|_{z_1=e^{j\omega_1 T},z_2=e^{j\omega_2 T},z_3=e^{j\omega_3 T}} \tag{5.2}$$

where

$$H_D(z_1,z_2,z_3)H_D(z_1^{-1},z_2^{-1},z_3^{-1})=\left(\sum_{l=0}^{N}\sum_{m=0}^{N}\sum_{n=0}^{N}a(l,m,n)z_1^{l}z_2^{m}z_3^{n}\right)\left(\sum_{l=0}^{N}\sum_{m=0}^{N}\sum_{n=0}^{N}a(l,m,n)z_1^{-l}z_2^{-m}z_3^{-n}\right)$$

$$=\left(\sum_{l=-N}^{N}\sum_{m=-N}^{N}\sum_{n=-N}^{N}c(l,m,n)z_1^{-l}z_2^{-m}z_3^{-n}\right) \tag{5.3}$$

in which the new coefficients *c(l,m,n)* are functions of *a(l,m,n)* and related through 16-hydral symmetry defined by equation (1.19b).

The magnitude-squared function is obtained by evaluating $H_d(z_1,z_2,z_3)$ and $H_d(z_1^{-1},z_2^{-1},z_3^{-1})$ on the unite sphere and is of the form.

(5.7) can be expressed as a set

$$M_D^2(\omega_1,\omega_2,\omega_3)=\left|H_D(e^{j\omega_1 T},e^{j\omega_2 T},e^{j\omega_3 T})\right|^2 = N_1(j\omega_1,j\omega_2,j\omega_3)$$
$$= c(0,0,0)+\sum_{l=1}^{N}\sum_{m=1}^{N}\sum_{n=1}^{N}2c(l,m,n)\cos(l\omega_1)\cos(m\omega_2)\cos(n\omega_3) \quad (5.4)$$

It is seen from equation (5.6) that $N_1(j\omega_1,j\omega_2,j\omega_3)$ is linear combination of *c(l,m,n)*.

Now, given the desired magnitude-squared $M_I(j\omega_1,j\omega_2,j\omega_3)$, then the approximation problem is that of finding the filter coefficients such that

$$-\varepsilon(\omega_1,\omega_2,\omega_3)\le N_1(j\omega_1,j\omega_2,j\omega_3)-M_I(\omega_1,\omega_2,\omega_3)\le\varepsilon(\omega_1,\omega_2,\omega_3) \quad (5.5)$$

Where $\varepsilon(\omega_1,\omega_2,\omega_3)$ is the tolerance function.

equation (5.5) can be expressed as a set of linear inequalities in the coefficient *c(l,m,n)* as follows

$$N_1(j\omega_1,j\omega_2,j\omega_3)\le\varepsilon(\omega_1,\omega_2,\omega_3)D_I(j\omega_1,j\omega_2,j\omega_3)$$
$$-N_1(j\omega_1,j\omega_2,j\omega_3)\le\varepsilon(\omega_1,\omega_2,\omega_3)D_I(j\omega_1,j\omega_2,j\omega_3 \quad (5.6a)$$

with the additional inequality

$$-N(j\omega_1,j\omega_2,j\omega_3)\le 0 \quad (5.6b)$$

The approximation problem is thus defined. To solve the system of linear inequalities(5.6a) (5.6b), a dummy variable δ is subtracted from the left side of each constraint in the system, and this variable is then minimized. If the resulting value of δ is 0, then a solution exists to the approximation problem.

Design Example 5.1

We which to design a 7 ×7 ×7 3-D FIR filter with the following specifications

$$H(e^{-j\omega_1}, e^{-j\omega_2}, e^{-j\omega_3}) = \begin{cases} 1 & \text{for } 0 \le \sqrt{\omega_1^2 + \omega_2^2 + \omega_3^2} \le 1 \\ \\ 0 & \text{for } 2 \le \sqrt{\omega_1^2 + \omega_2^2 + \omega_3^2} \le \pi \end{cases}$$

Using cubic , diagonal and 16-hydral symmetries, The number of coefficients is reduced from 343 to 20. Table 5.1 shows the 20 coefficients obtained and figure 5.1,5.2,5.3 and 5.4 show the magnitude response of the designed filter at different values of ω_3.

Table 5-1: The coefficients of the 7 x 7 x 7 3-D high-pass FIR filter .

A(0,0,0)	0.3321472E-01	A(3,0,0)	0.1479945
A(1,0,0)	0.6026059E-01	A(3,1.0)	0.5402668E-01
A(1,1,0)	0.3360325E-01	A(3,1,1)	0.8054069E-02
A(1,1,1)	0.1079219E-01	A(3,2,0)	-0.7039470E-02
A(2,0,0)	0.9833931E-01	A(3,2,1)	-0.1660429E-01
A(2,1,0)	.4632999E-01	A(3,2,2)	-0.1314769E-01
A(2,1,1)	0.1266424E-01	A(3,3,0)	-0.4153535E-01
A(2,2,0)	0.1078081E-01	A(3,3,1)	-0.2829976E-01
A(2,2,1)	-0.1639693E-02	A(3,3,2)	-0.1591307E-01
A(2,2,2)	-0.3592633E-02	A(3,3,3)	-0.7768183E-02

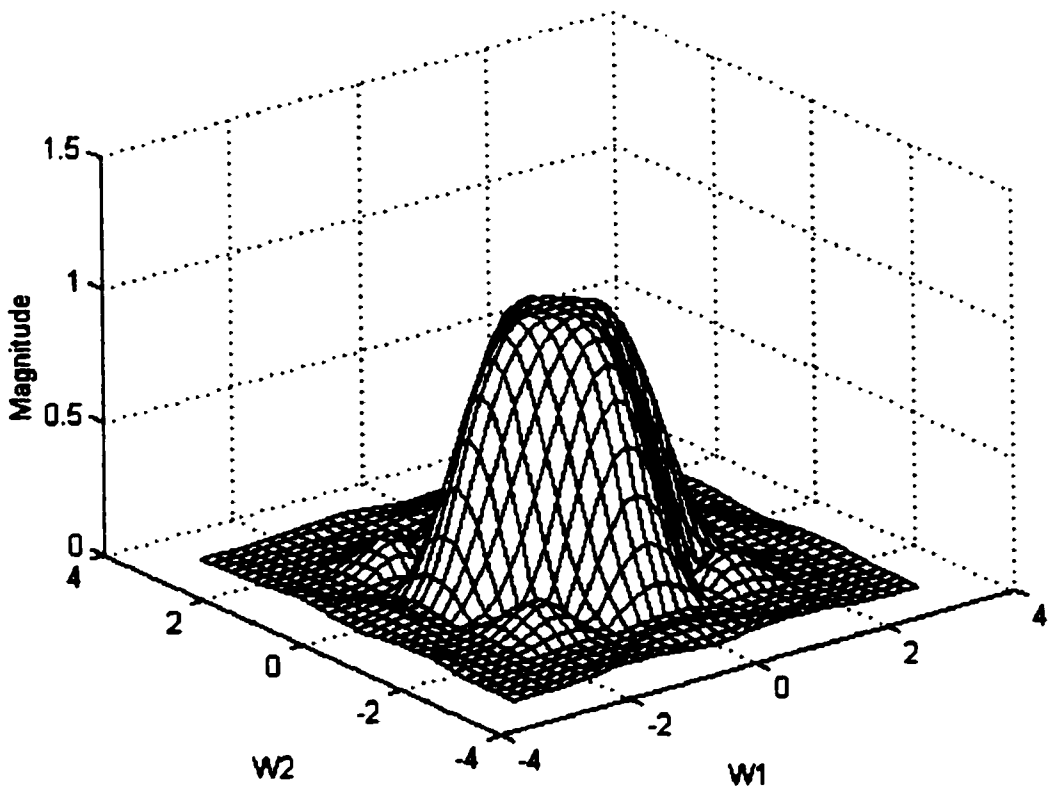

Fig. 5.1 Magnitude response of example 5.1 at $\omega_3 = 0$

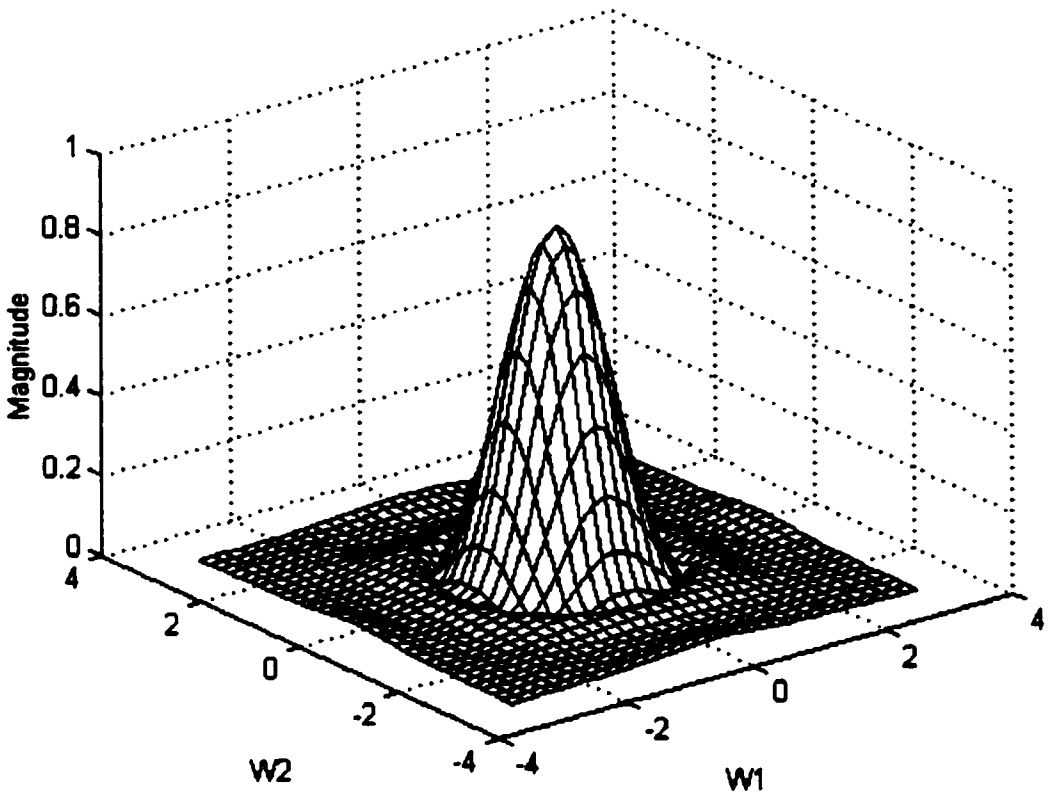

Fig. 5.2 Magnitude response of example 5.1 at $\omega_3 = 0.9817$

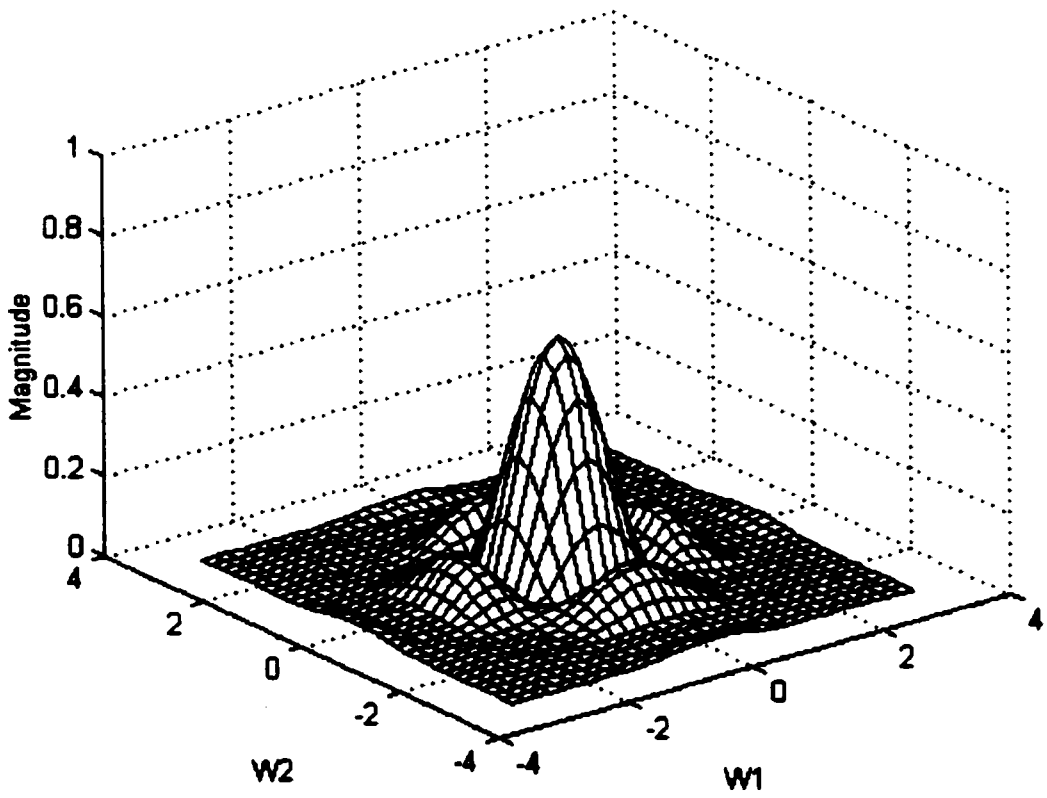

Fig. 5.3 Magnitude response of example 5.1 at $\omega_3 = 1.3744$

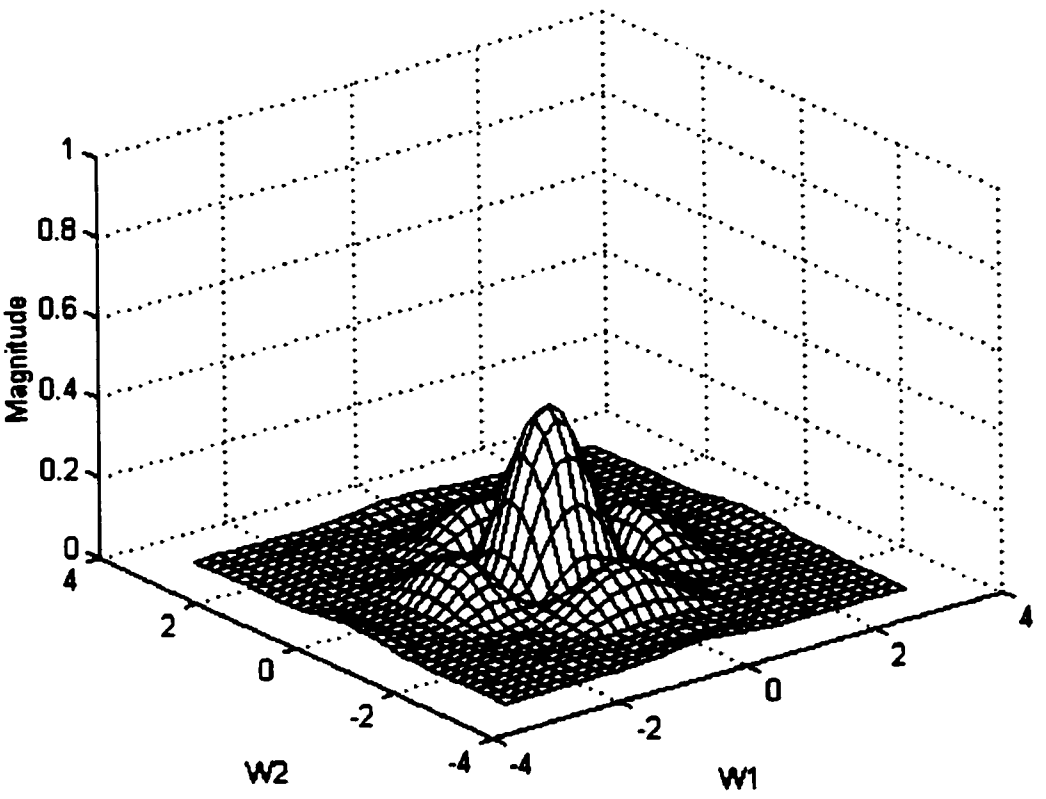

Fig. 5.4 Magnitude response of example 5.1 at $\omega_3 = 1.5708$

5.4 Design of 3-D IIR Filters using linear programming

We next utilize linear programming to design of 3-D recursive digital filters satisfying a prescribed magnitude and group delay specifications. The method to be presented here is an extension of the 2-D design approach described by Chottera and Jullien in[37][50]. The approach deals with the general class of 3-D IIR filters as well as a special case of filters namely separable nominator non-separable denominator transfer functions.

The proposed method is iterative in nature and uses linear optimization techniques to determine the coefficients of the filter. Suitable stability constraints on filter coefficients are indicated for stable design and an algorithm for designing near-linear phase is described.

5.4.1 Theory of approximation

let $H(z_1,z_2,z_3)$ be the transfer function of a 3-D recursive digital filter and has the following form

$$H(z_1,z_2,z_3)=\frac{A(z_1,z_2,z_3)}{B(z_1,z_2,z_3)}=\frac{\sum_{i=0}^{N}\sum_{j=0}^{N}\sum_{k=0}^{N}a(i,j,k)z_1^{-i}z_2^{-j}z_3^{-k}}{\sum_{i=0}^{N}\sum_{j=0}^{N}\sum_{\substack{k=0\\ i+j+k\neq 0}}^{N}b(i,j,k)z_1^{-i}z_2^{-j}z_3^{-k}} \qquad (5.7)$$

Where

$z_1=e^{-j\omega_1}$, $z_2=e^{-j\omega_2}$ and $z_3=e^{-j\omega_3}$

ω_1,ω_2 and ω_3 are the normalized frequency variables

Without any lose of generality the term b(0,0,0) in Eq.(6.1) can be set equal to 1.0.

Given an arbitrary magnitude and phase response specifications, it is desired to formulate the digital filter transfer function into a linear programming problem such that the constraints of the linear programming are in term of the filter specification.

Let $R(\omega_{1l},\omega_{2m},\omega_{3n})$ and $\phi(\omega_{1l},\omega_{2m},\omega_{3n})$ be the given magnitude and phase specification respectively, specified over a frequency grid ω_{1l},ω_{2m} and ω_{3n}; i=1... L_1 j=1... L_2 and k=1... L_3.

The real and imaginary components of the frequency domain specifications can be written as

$$H_R(\omega_{1l},\omega_{2m},\omega_{3n}) = R(\omega_{1l},\omega_{2m},\omega_{3n})\left[\cos(\phi(\omega_{1l},\omega_{2m},\omega_{3n}))\right] \tag{5.8}$$

$$H_I(\omega_{1l},\omega_{2m},\omega_{3n}) = R(\omega_{1l},\omega_{2m},\omega_{3n})\left[\sin(\phi(\omega_{1l},\omega_{2m},\omega_{3n}))\right] \tag{5.9}$$

The problem of approximating the characteristics of recursive filter can be formulated into a linear programming by defining the error between the desired frequency and the approximated frequency as follows

$$\varepsilon(\omega_{1l},\omega_{2m},\omega_{3n}) = \left[H_R(\omega_{1l},\omega_{2m},\omega_{3n}) + jH_I(\omega_{1l},\omega_{2m},\omega_{3n})\right] - \left[\frac{A(e^{-j\omega_{1l}},e^{-j\omega_{2m}},e^{-j\omega_{3n}})}{B(e^{-j\omega_{1l}},e^{-j\omega_{2m}},e^{-j\omega_{3n}})}\right] \tag{5.10}$$

multiplying both sides by $B(\omega_{1i},\omega_{2j},\omega_{3k})$ a weighted error function (which is linear in term of filter coefficients) is obtained and is given by

$$\begin{aligned} &B(e^{-j\omega_{1l}},e^{-j\omega_{2m}},e^{-j\omega_{3n}})\,\varepsilon(\omega_{1l},\omega_{2m},\omega_{3n}) = \\ &B(e^{-j\omega_{1l}},e^{-j\omega_{2m}},e^{-j\omega_{3n}})\left[H_R(\omega_{1l},\omega_{2m},\omega_{3n}) + jH_I(\omega_{1l},\omega_{2m},\omega_{3n})\right] - \left[A(e^{-j\omega_{1l}},e^{-j\omega_{2m}},e^{-j\omega_{3n}})\right] \end{aligned} \tag{5.11}$$

The quantities in Eq.(5.11) can be separated into their real and imaginary components as:

$$B(e^{-j\omega_{1l}}, e^{-j\omega_{2m}}, e^{-j\omega_{3n}})\ \varepsilon(\omega_{1l}, \omega_{2m}, \omega_{3n}) = E_R(\omega_{1l}, \omega_{2m}, \omega_{3n}) + j\,E_I(\omega_{1l}, \omega_{2m}, \omega_{3n}) \tag{5.12}$$

$$A(e^{-j\omega_{1l}}, e^{-j\omega_{2m}}, e^{-j\omega_{3n}}) = A_R(\omega_{1l}, \omega_{2m}, \omega_{3n}) - j\,A_I(\omega_{1l}, \omega_{2m}, \omega_{3n}) \tag{5.13}$$

$$B(e^{-j\omega_{1l}}, e^{-j\omega_{2m}}, e^{-j\omega_{3n}}) = B_R(\omega_{1l}, \omega_{2m}, \omega_{3n}) - j\,B_I(\omega_{1l}, \omega_{2m}, \omega_{3n}) \tag{5.14}$$

Substituting in the original expression

$$\begin{aligned} E_R(\omega_{1l}, \omega_{2m}, \omega_{3n}) + j\,E_I(\omega_{1l}, \omega_{2m}, \omega_{3n}) = {} & H_R(\omega_{1l}, \omega_{2m}, \omega_{3n})\left[B_R(\omega_{1l}, \omega_{2m}, \omega_{3n}) - jB_I(\omega_{1l}, \omega_{2m}, \omega_{3n})\right] \\ & + j\,H_I(\omega_{1l}, \omega_{2m}, \omega_{3n})\left[B_R(\omega_{1l}, \omega_{2m}, \omega_{3n}) - JB_I(\omega_{1l}, \omega_{2m}, \omega_{3n})\right] \\ & - \left[A_R(\omega_{1l}, \omega_{2m}, \omega_{3n}) - j\,A_I(\omega_{1l}, \omega_{2m}, \omega_{3n})\right] \end{aligned} \tag{5.15}$$

Equating the real and imaginary parts results in

The real part

$$\begin{aligned} E_R(\omega_{1l}, \omega_{2m}, \omega_{3n}) = {} & H_R(\omega_{1l}, \omega_{2m}, \omega_{3n})[B_R(\omega_{1l}, \omega_{2m}, \omega_{3n})] + H_I(\omega_{1l}, \omega_{2m}, \omega_{3n})[B_I(\omega_{1l}, \omega_{2m}, \omega_{3n})] \\ & - A_R(\omega_{1l}, \omega_{2m}, \omega_{3n}) \end{aligned} \tag{5.16}$$

The imaginary part

$$\begin{aligned} E_I(\omega_{1l}, \omega_{2m}, \omega_{3n}) = {} & H_I(\omega_{1l}, \omega_{2m}, \omega_{3n})[B_R(\omega_{1l}, \omega_{2m}, \omega_{3n})] - H_I(\omega_{1l}, \omega_{2m}, \omega_{3n})[B_I(\omega_{1l}, \omega_{2m}, \omega_{3n})] \\ & + j\,[A_R(\omega_{1l}, \omega_{2m}, \omega_{3n})] \end{aligned} \tag{5.17}$$

Now it is possible to put the filter equations into linear programming problem such that if we choose a quantity ε such that

$$\left|E_R(\omega_{1l},\omega_{2m},\omega_{3n})\right| \le \varepsilon \tag{5.18}$$

and

$$\left|E_I(\omega_{1l},\omega_{2m},\omega_{3n})\right| \le \varepsilon \tag{5.19}$$

Then we can minimize ε subject to

$$\begin{aligned} H_R(\omega_{1l},\omega_{2m},\omega_{3n})\, B_R(\omega_{1l},\omega_{2m},\omega_{3n}) + H_I(\omega_{1l},\omega_{2m},\omega_{3n})\, B_I(\omega_{1l},\omega_{2m},\omega_{3n}) \\ - A_R(\omega_{1l},\omega_{2m},\omega_{3n}) + \varepsilon \le 0 \end{aligned} \tag{5.20}$$

$$\begin{aligned} H_I(\omega_{1l},\omega_{2m},\omega_{3n})\, B_R(\omega_{1l},\omega_{2m},\omega_{3n}) - H_R(\omega_{1l},\omega_{2m},\omega_{3n})\, B_I(\omega_{1l},\omega_{2m},\omega_{3n}) \\ + A_I(\omega_{1l},\omega_{2m},\omega_{3n}) + \varepsilon \le 0 \end{aligned} \tag{5.21}$$

$$\begin{aligned} -H_R(\omega_{1l},\omega_{2m},\omega_{3n})\, B_R(\omega_{1l},\omega_{2m},\omega_{3n}) - H_I(\omega_{1l},\omega_{2m},\omega_{3n})\, B_I(\omega_{1l},\omega_{2m},\omega_{3n}) \\ + A_R(\omega_{1l},\omega_{2m},\omega_{3n}) + \varepsilon \le 0 \end{aligned} \tag{5.22}$$

$$\begin{aligned} -H_I(\omega_{1l},\omega_{2m},\omega_{3n})\, B_R(\omega_{1l},\omega_{2m},\omega_{3n}) + H_R(\omega_{1l},\omega_{2m},\omega_{3n})\, B_I(\omega_{1l},\omega_{2m},\omega_{3n}) \\ - A_I(\omega_{1l},\omega_{2m},\omega_{3n}) + \varepsilon \le 0 \end{aligned} \tag{5.23}$$

5.4.2 Stability constraints

If the designed filter is desired to be stable, however additional constraints must be added. Since the approximation procedure involves linear programming , stability constraints mus in linear format. To insure the stability of 3-D recursive filter the real part of the denominator of the transfer function must not equal to zero on the unite sphere. The actual constraints are slightly modified and can be written as

$\mathrm{RE}(B(Z_1,Z_2,Z_3)) \geq \Delta C$ For $|Z_1|=1$, $|Z_2|=1$ and $|Z_3|=1$, where ΔC is a small positive quantity. Since

$$RE(B(Z_1,Z_2,Z_3)) = \sum_{l=0}^{N}\sum_{m=0}^{N}\sum_{n=0}^{N} b(l,m,n)\left[\cos(l\omega_1)\cos(m\omega_2)\cos(n\omega_3)\right] \quad (5.24)$$

Then the stability constraints could be written as

$$1+\sum_{l=0}^{N}\sum_{m=0}^{N}\sum_{\substack{n=0 \\ l+m+n\neq 0}}^{N} b(l,m,n)\left[\cos(l\omega_1)\cos(m\omega_2)\cos(n\omega_3)\right] \geq \Delta C \quad (5.25)$$

or

$$-\sum_{l=0}^{N}\sum_{m=0}^{N}\sum_{n=0}^{N} b(l,m,n)\left[\cos(l\omega_1)\cos(m\omega_2)\cos(n\omega_3)\right] \leq 1-\Delta C \quad (5.26)$$

It should be noted that the constraints of (5.26) is not a general stability constraints; however, it is sufficient condition for $B(\omega_1,\omega_2,\omega_3)$ to be a stable polynomial.

5.4.3 Design Procedure

We now use the linear programming approach to design a stable, linear phase 3-D recursive filter. The desired linear phase characteristics can be specified in term of spatial delay. Let $\phi(\omega_1,\omega_2,\omega_3)$, be the phase characteristics and $\tau(\omega_1,\omega_2,\omega_3)$ be the delay characteristics. The delay is given by

$$\tau_1(\omega_1,\omega_2,\omega_3) = -\frac{\partial\phi(\omega_1,\omega_2,\omega_3)}{\partial\omega_1} \tag{5.27}$$

$$\tau_2(\omega_1,\omega_2,\omega_3) = -\frac{\partial\phi(\omega_1,\omega_2,\omega_3)}{\partial\omega_2} \tag{5.28}$$

$$\tau_3(\omega_1,\omega_2,\omega_3) = -\frac{\partial\phi(\omega_1,\omega_2,\omega_3)}{\partial\omega_3} \tag{5.29}$$

If the desired phase characteristics is desired to be linear, then $\tau_1(\omega_1,\omega_2,\omega_3)$, $\tau_2(\omega_1,\omega_2,\omega_3)$ and $\tau_3(\omega_1,\omega_2,\omega_3)$ are assigned constants values. By setting $\tau_1(\omega_1,\omega_2,\omega_3) = \tau_x$, $\tau_2(\omega_1,\omega_2,\omega_3) = \tau_y$ and $\tau_3(\omega_1,\omega_2,\omega_3) = \tau_z$, the desired linear phase characteristics can be written as:

$$\phi(\omega_1,\omega_2,\omega_3) = -(\tau_x\omega_1 + \tau_y\omega_2 + \tau_z\omega_3) \tag{5.30}$$

This specification can be further simplified by setting $\tau_x = \tau_y = \tau_z = \tau_c$; indicating

the same amount of delay in the three directions in the spatial domain. The phase characteristics is then given by

$$\phi(\omega_1,\omega_2,\omega_3) = -\tau_c(\omega_1+\omega_2+\omega_3) \tag{5.31}$$

The real and imaginary parts of the desired frequency response are then written as

$$H_R(\omega_{1i},\omega_{2j},\omega_{3k}) = R(\omega_{1i},\omega_{2j},\omega_{3k})\left[\cos(-\tau_c(\omega_{1i},\omega_{2j},\omega_{3k}))\right] \tag{5.32}$$

$$H_I(\omega_{1i},\omega_{2j},\omega_{3k}) = R(\omega_{1i},\omega_{2j},\omega_{3k})\left[\sin(-\tau_c(\omega_{1i},\omega_{2j},\omega_{3k}))\right] \tag{5.33}$$

Now with

$$A_R(\omega_{1i},\omega_{2j},\omega_{3k}) = \sum_{l=0}^{N}\sum_{m=0}^{N}\sum_{n=0}^{N} a(l,m,n)\cos(l\omega_{1i}+m\omega_{2j}+n\omega_{3k}) \tag{5.34}$$

$$A_I(\omega_{1i},\omega_{2j},\omega_{3k}) = \sum_{l=0}^{N}\sum_{m=0}^{N}\sum_{n=0}^{N} a(l,m,n)\sin(l\omega_{1i}+m\omega_{2j}+n\omega_{3k}) \tag{5.35}$$

$$B_R(\omega_{1i},\omega_{2j},\omega_{3k}) = 1+\sum_{\substack{l=0\\ l+m+n\neq 0}}^{N}\sum_{m=0}^{N}\sum_{n=0}^{N} b(l,m,n)\cos(l\omega_{1i}+m\omega_{2j}+n\omega_{3k}) \tag{5.36}$$

$$B_I(\omega_{1i},\omega_{2j},\omega_{3k}) = \sum_{\substack{l=0\\ l+m+n\neq 0}}^{N}\sum_{m=0}^{N}\sum_{n=0}^{N} b(l,m,n)\sin(l\omega_{1i}+m\omega_{2j}+n\omega_{3k}) \tag{5.37}$$

The linear programming problem becomes :

Minimize ε

Subject to

$$\sum_{\substack{l=0\\ l+m+n\neq 0}}^{N}\sum_{m=0}^{N}\sum_{n=0}^{N} b(l,m,n)\, R(\omega_{1i},\omega_{2j},\omega_{3k})\cos((l-\tau_c)\omega_{1i}+(m-\tau_c)\omega_{2j}+(n-\tau_c)\omega_{3k})-$$

$$\sum_{l=0}^{N}\sum_{m=0}^{N}\sum_{n=0}^{N} a(l,m,n)\cos(l\omega_{1i}+m\omega_{2j}+n\omega_{3k})-\varepsilon+R(\omega_{1i},\omega_{2j},\omega_{3k})\cos(-\tau_c(\omega_{1i}+\omega_{2j}+\omega_{3k}))$$

$$\leq 0 \qquad (5.38)$$

$$\sum_{\substack{l=0\\ l+m+n\neq 0}}^{N}\sum_{m=0}^{N}\sum_{n=0}^{N} b(l,m,n)\, R(\omega_{1i},\omega_{2j},\omega_{3k})-\sin((l+\tau_c)\omega_{1i}+(m+\tau_c)\omega_{2j}+(n+\tau_c)\omega_{3k})$$

$$+\sum_{l=0}^{N}\sum_{m=0}^{N}\sum_{n=0}^{N} a(l,m,n)\sin(l\omega_{1i}+m\omega_{2j}+n\omega_{3k})+R(\omega_{1i},\omega_{2j},\omega_{3k})\sin s(-\tau_c(\omega_{1i}+\omega_{2j}+\omega_{3k}))$$

$$-\varepsilon+\leq 0 \qquad (5.39)$$

$$\sum_{\substack{l=0\\ l+m+n\neq 0}}^{N}\sum_{m=0}^{N}\sum_{n=0}^{N} b(l,m,n)\, R(\omega_{1i},\omega_{2j},\omega_{3k})\cos((l-\tau_c)\omega_{1i}+(m-\tau_c)\omega_{2j}+(n-\tau_c)\omega_{3k})$$

$$+\sum_{l=0}^{N}\sum_{m=0}^{N}\sum_{n=0}^{N} a(l,m,n)\cos(l\omega_{1i}+m\omega_{2j}+n\omega_{3k})-R(\omega_{1i},\omega_{2j},\omega_{3k})\cos(-\tau_c(\omega_{1i}+\omega_{2j}+\omega_{3k}))$$

$$-\varepsilon+\leq 0 \qquad (5.40)$$

$$-\sum_{\substack{l=0\\ l+m+n\neq 0}}^{N}\sum_{m=0}^{N}\sum_{n=0}^{N} b(l,m,n)\, R(\omega_{1i},\omega_{2j},\omega_{3k}) - \sin((l-\tau_c)\omega_{1i} + (m-\tau_c)\omega_{2j} + (n-\tau_c)\omega_{3k})$$

$$+\sum_{l=0}^{N}\sum_{m=0}^{N}\sum_{n=0}^{N} a(l,m,n)\sin(l\omega_{1i} + m\omega_{2j} + n\omega_{3k}) - R(\omega_{1i},\omega_{2j},\omega_{3k})\sin(-\tau_c(\omega_{1i} + \omega_{2j} + \omega_{3k}))$$

$$-\varepsilon + \leq 0 \tag{5.41}$$

and the stability constraints

$$-\sum_{\substack{l=0\\ l+m+n\neq 0}}^{N}\sum_{m=0}^{N}\sum_{n=0}^{N} b(l,m,n)\cos(l\omega_1 + m\omega_2 + n\omega_3) \leq 1 - \Delta C \tag{5.42}$$

where Δc is a small positive number.

The design procedure can be described as follows

1- Specify the desired magnitude characteristics:

$R(\omega_{1i},\omega_{2j},\omega_{3k})$

for $i = 1,2,\ldots,L_1$, $j = 1,2,\ldots,L_2$; $k = 1,2,\ldots,L_3$

2- Choose the order of the filter say N; and choose a range for τ_c.

3- Solve the linear programming problem and changing the value of τ_c until ε is minimized.

4- Choose the coefficients of the filters for which ε is minimum and compute the frequency response.

Design Example 6.1

We wish to design 1x1x1 stable IIR filter with the following specifications using the linear programming approach outlined previously

$$\left|H(e^{-j\omega_1},e^{-j\omega_2},e^{-j\omega_3})\right| = \begin{cases} 1 \quad \text{for} \quad 0 \le \sqrt{\omega_1^2+\omega_2^2+\omega_3^2} \le 1.0 \\ 0 \quad \text{for} \quad 2 \le \sqrt{\omega_1^2+\omega_2^2+\omega_3^2} \le \pi \end{cases}$$

Using the procedure outlined table 5.3 shows the coefficients obtained and figures 5.5-5.8 show the magnitude response at different values of ω_3.

a coefficients	b coefficients
-0.0458	1.0000
-0.0475	-0.4795
-0.0475	-0.4795
-0.0549	0.2589
-0.0475	-0.4795
-0.0549	0.2589
-0.0549	0.2589
-0.0384	-0.0384

Table 5-2 IIR filter coefficients

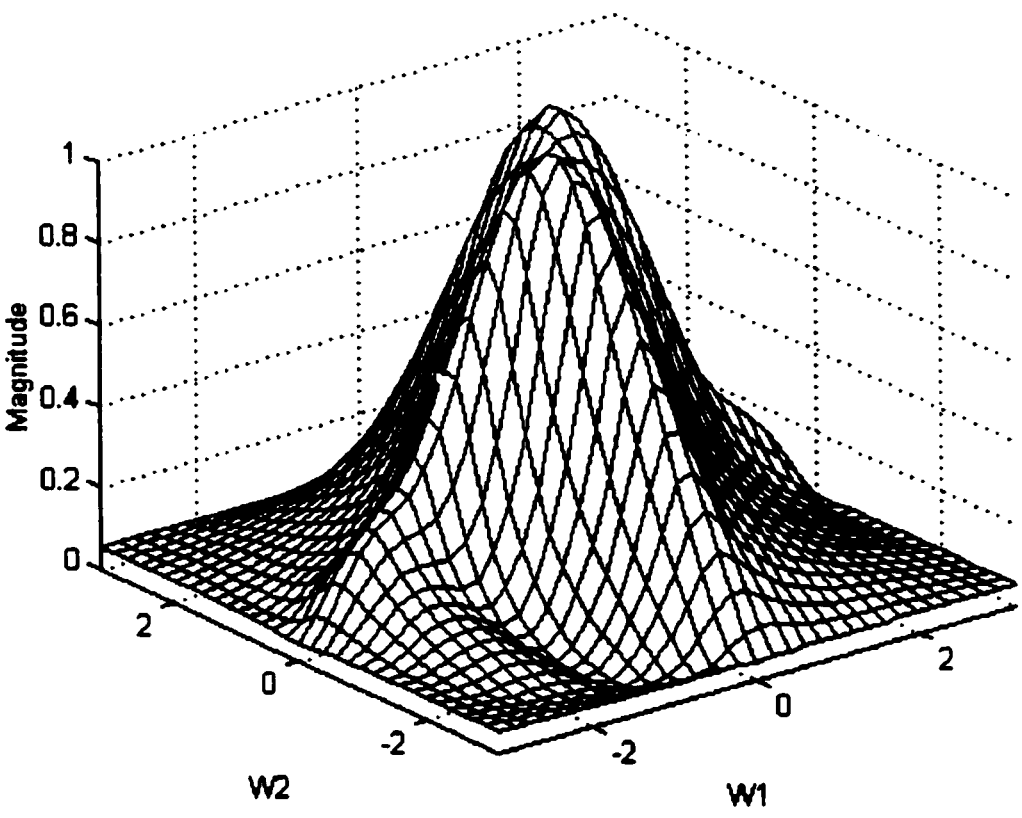

Fig.5.5 Magnitude response of 1x1x1 IIR filter with ω_c=2.0 at ω_3=0

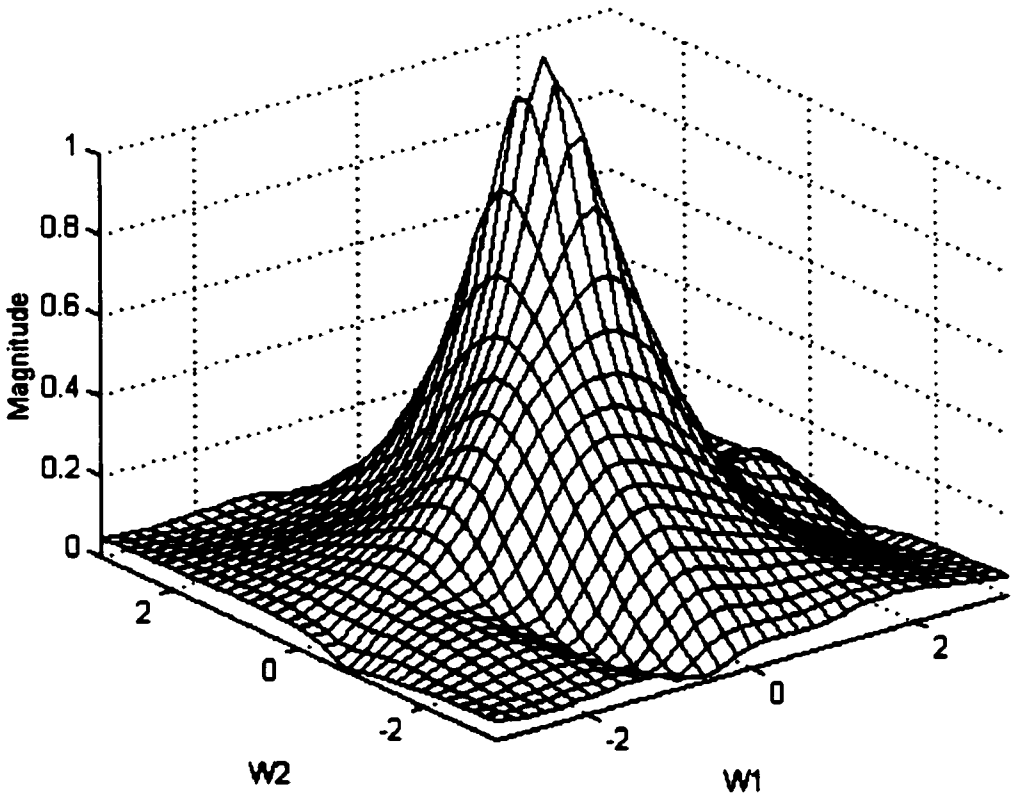

Fig.5.6 Magnitude response of 1x1x1 IIR filter with ω_c=2.0 at ω_3=0.9817

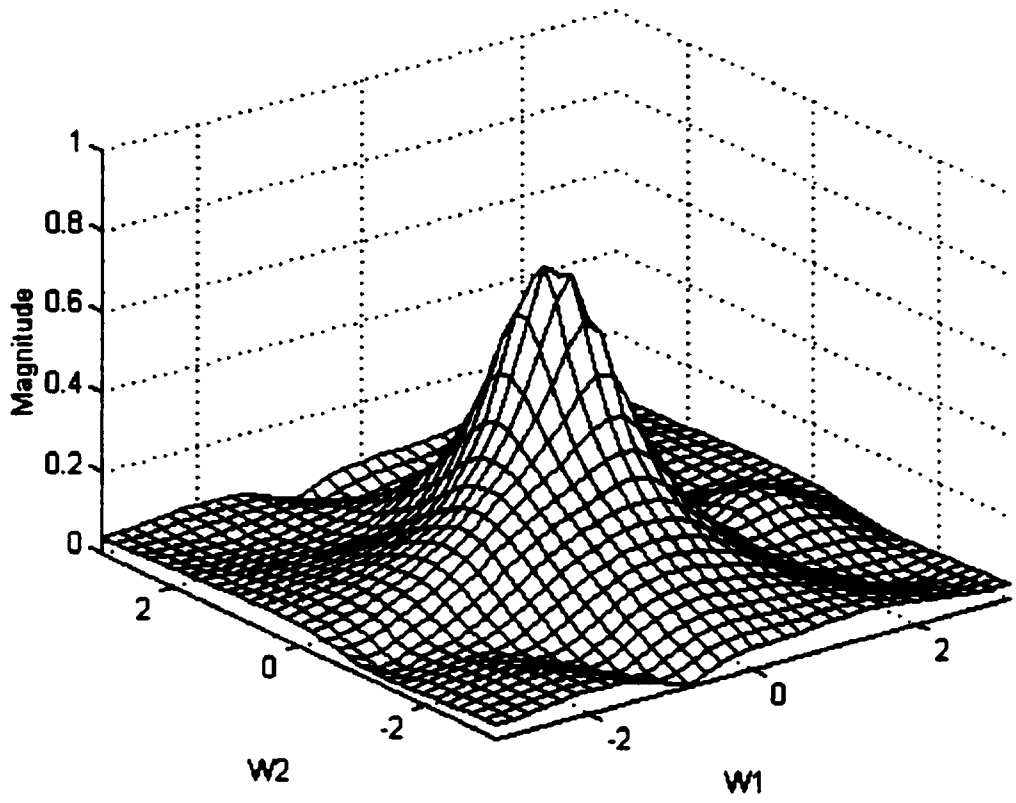

Fig.5.7 Magnitude response of 1x1x1 IIR filter with ω_c =2.0 at ω_3 =1.9635

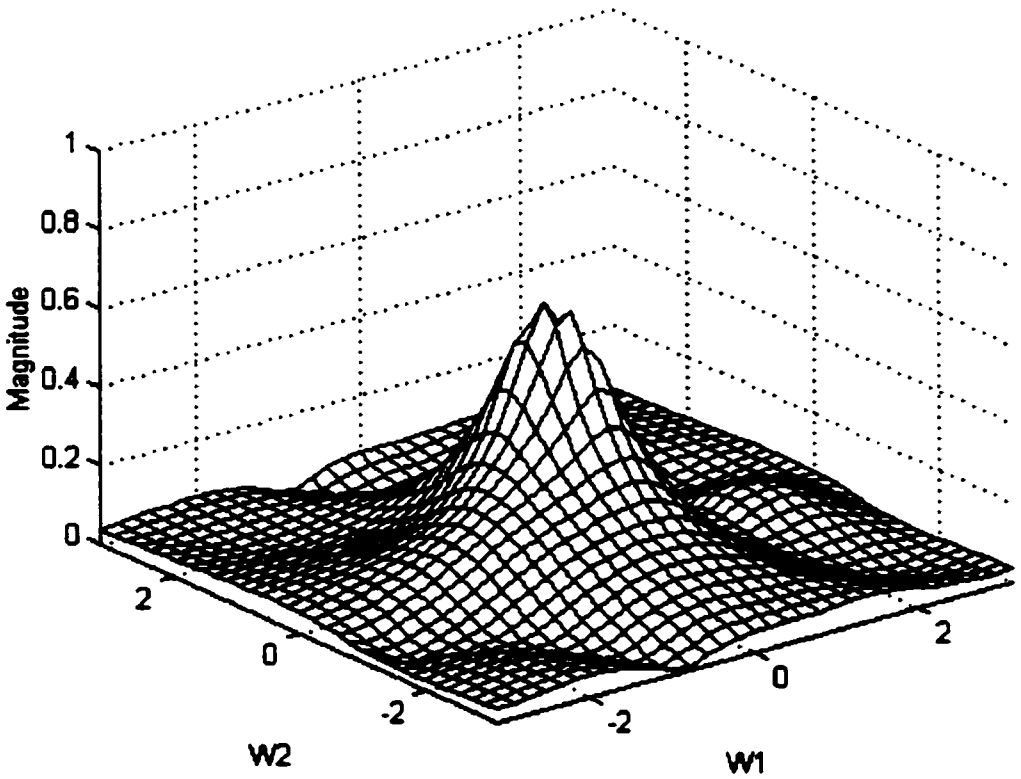

Fig.5.8 Magnitude response of 1x1x1 IIR filter with ω_c=2.0 at ω_3=2.3562

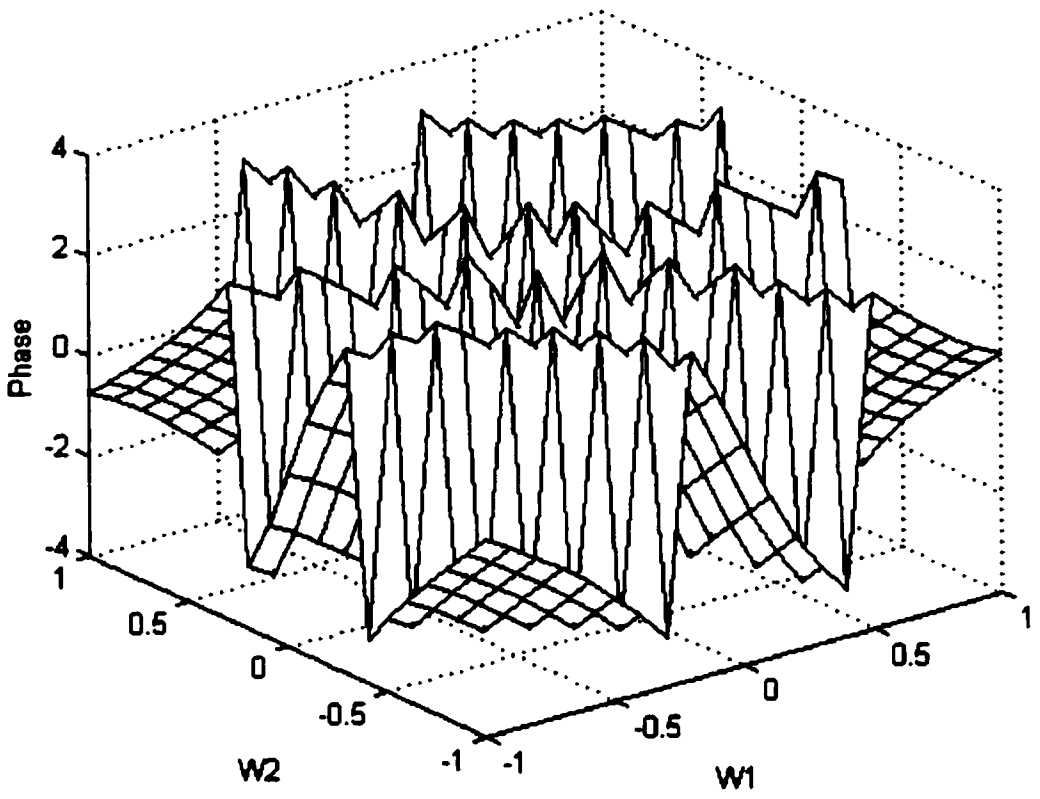

Fig.5.9 Phase response of 1x1x1 IIR filter at ω_3 =0

5.5 DESIGN OF 3-D RECURSIVE FILTERS WITH SEPARABLE AND NUMERATOR NON- SEPARABLE DENOMINATOR

Recently many researchers have focussed on the design of 3-D IIR filters with separable numerator and non-separable denominator transfer function [27,29,30]. In order to reduce the computation burden the design process is broken into two steps. In the first step, the coefficients of the denominator are determined by designing three all- poles 1-D digital filters. by cascading the three 1-D filters a 3-D filter with a rectangular cutoff boundary is generated. In the second step the resultant is cascaded with a 3-D FIR filter having 16-hydral symmetry. The transfer function is optimized using linear optimization to determine the filter coefficients. The transfer function of separable denominator filters has the following form:

$$H(z_1,z_2,z_3)=\frac{\sum_{i=0}^{N}\sum_{j=0}^{N}\sum_{k=0}^{N}a(i,j,k)z_1^{-i}z_2^{-j}z_3^{-k}}{\left(\sum_{i=0}^{N}a(i)z_1^{-i}\right).\left(\sum_{j=0}^{N}a(j)z_2^{-j}\right).\left(\sum_{k=0}^{N}a(k)z_3^{-k}\right)} \qquad (5.43)$$

$$=\frac{A(z_1,z_2,z_3)}{B(z_1).B(z_2).B(z_3)}$$

To ensure the stability of the filter

$$B_i(z_i)\neq 0 \qquad |z_i|\geq 1 \quad \text{for i} = 1,2,3 \qquad (5.44)$$

A 1-D all poles digital filter has the following form

$$H_i(z_i)=\frac{1}{B_i(z_i)} \qquad (5.45)$$

Cascading the three 1-D digital filters together a 3-D filter with a rectangular cut off boundary

is generated as follows:

$$H_1(z_1,z_2,z_3) = H_1(z_1).H_2(z_2).H_3(z_3) \tag{5.46}$$

Now $H_1(z_1,z_2,z_3)$ is cascaded by a 3-D FIR filters with cubic, diagonal and 26-hydral symmetry which is written as

$$H_2(z_1,z_2,z_3) = \sum_{i=0}^{\frac{N-1}{2}} \sum_{j=i}^{\frac{N-1}{2}} \sum_{k=j}^{\frac{N-1}{2}} a'(i,j,k)\cos(i.\omega_1).\cos(j.\omega_2).\cos(k.\omega_3) \tag{5.47}$$

This will yield

$$H(z_1,z_2,z_3) = H_1(z_1,z_2,z_3).H_2(z_1,z_2,z_3) \tag{5.48}$$

The magnitude error at different grid points along ω_1, ω_2 and ω_3 axis are calculated using

$$E_{Mag}(e^{-j\omega_1 l}, e^{-j\omega_2 m}, e^{-j\omega_3 n}, \psi) = \left|H_I(e^{-j\omega_1 l}, e^{-j\omega_2 m}, e^{-j\omega_3 n})\right| - \left|H_D(e^{-j\omega_1 l}, e^{-j\omega_2 m}, e^{-j\omega_3 n}, \psi)\right| \tag{5.49}$$

Where ψ is the coefficients vector and $|H_I|$ and $|H_D|$ are the magnitude response of the ideal and the designed filter respectively.

The error is calculated using the relationship

$$E_I(e^{j\omega_1 l}, e^{j\omega_2 m}, e^{j\omega_3 n}) = \sum_l \sum_m \sum_n E_{Mag}(e^{j\omega_1 l}, e^{j\omega_2 m}, e^{j\omega_3 n}, \psi) \tag{5.50}$$

It should be noted that the coefficients obtained in the first phase are constant in the second phase while $a'_{i,j,k}$ are determined in the second optimization phase. A substantial reduction in number of coefficients and frequency region can be achieved using the different symmetries.

To illustrate the usefulness of this method we present the following example

Example 5.3

We wish to design a low pass filter with separable denominator non-separable numerator that has the following specifications

$$\left|H(e^{j\omega_1},e^{j\omega_2},e^{j\omega_3})\right|=\begin{cases}1 & for\ \ 0\le\sqrt{\omega_1^2+\omega_2^2+\omega_3^2}\le 1\\ 0 & for\ \ 2\le\sqrt{\omega_1^2+\omega_2^2+\omega_3^2}\le\pi\end{cases}$$

Using 3 fourth order all poles filters the coefficients obtained are shown in table (6.3) .Figures 6.11-6.13 show the magnitude response of the designed filter at different values of ω_3.

Table 5.3 Coefficients of the filter of example 5.3

Numerator Coefficients		Denominator Coefficients		
$a_{0,0,0}$	0.6780302E-01	$a_{1,0}$ =3.6607	$a_{2,0}$ = 3.6607	$a_{3,0}$ = 3.6607
$a_{0,0,1}$	-0.9529408E-02	$a_{1,1}$ =-5.9370	$a_{2,1}$ = -5.9370	$a_{3,1}$ = -5.9370
$a_{0,0,2}$	0.4522154E-01			
$a_{0,1,1}$	0.1530579	$a_{1,2}$ = 5.4326	$a_{2,2}$ = 5.4326	$a_{3,2}$ =5.4326
$a_{0,1,2}$	0.2330837E-01	$a_{1,3}$ = -2.7059	$a_{2,3}$ = -2.7059	$a_{3,3}$ = -2.7059
$a_{0,2,2}$	0.1315885E-01	$a_{1,4}$ =0.5541	$a_{2,4}$ = 0.5541	$a_{3,4}$ =0.5541
$a_{1,1,1}$	0.7341378E-01			
$a_{1,1,2}$	0.7264757E-01			
$a_{1,2,2}$	0.6302781E-02			
$a_{2,2,2}$	-0.3077289E-02			

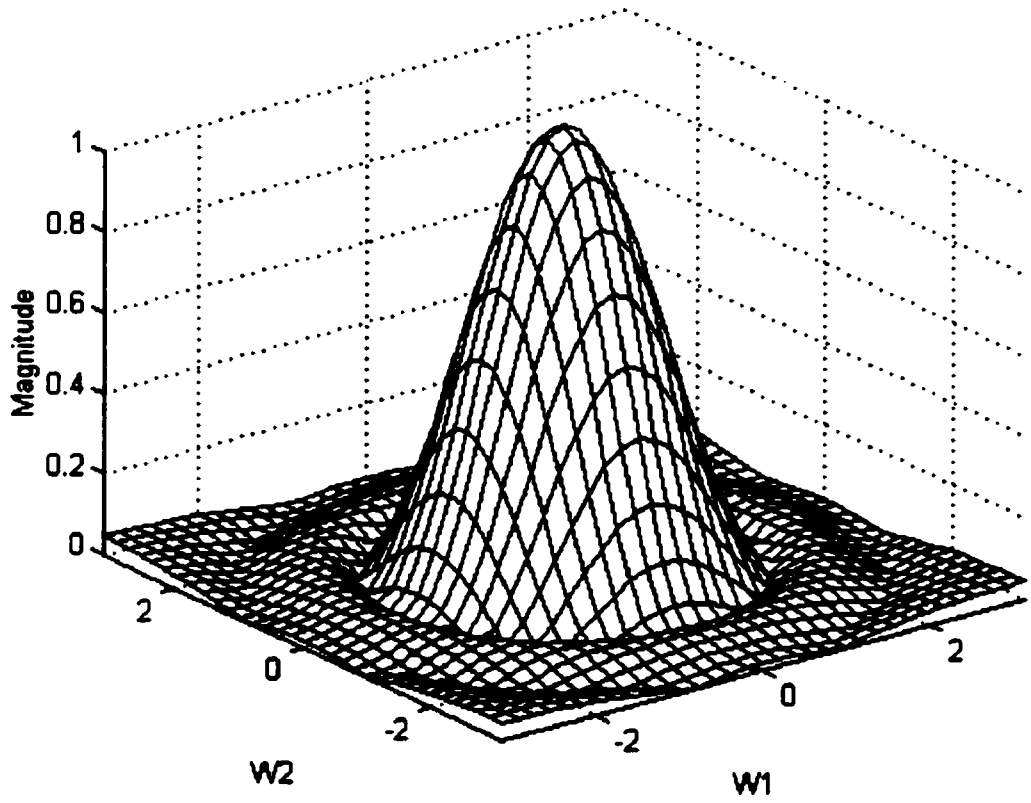

Figure 5.10 Magnitude response of the designed filter in example 5.3 at ω_3=0

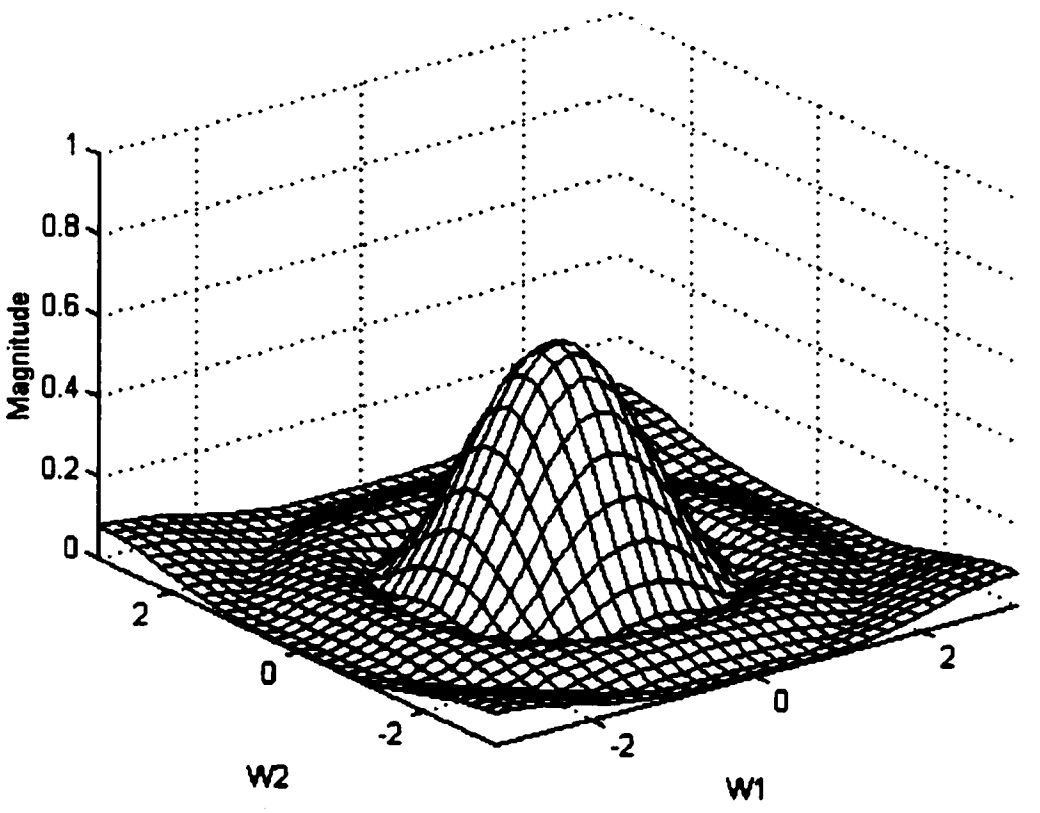

Figure 5.11 Magnitude response of the designed filter in example 5.3 at ω_3=0.9817

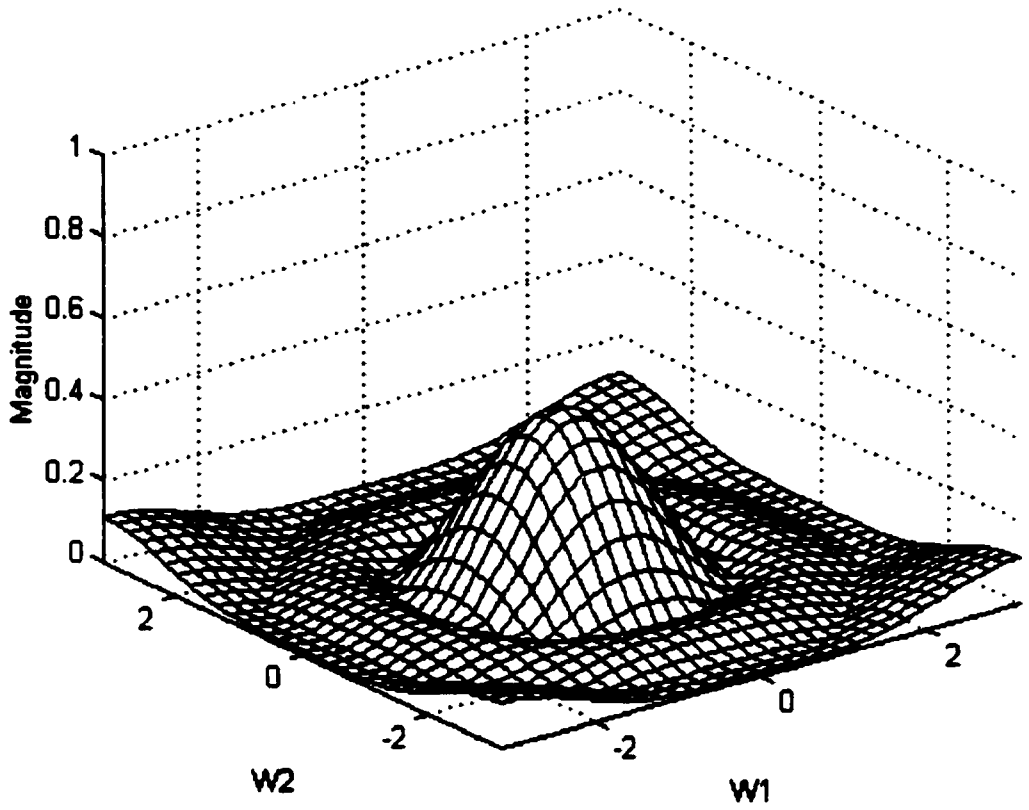

Figure 5.12 Magnitude response of the designed filter in example 5.3 at ω_3=1.1781

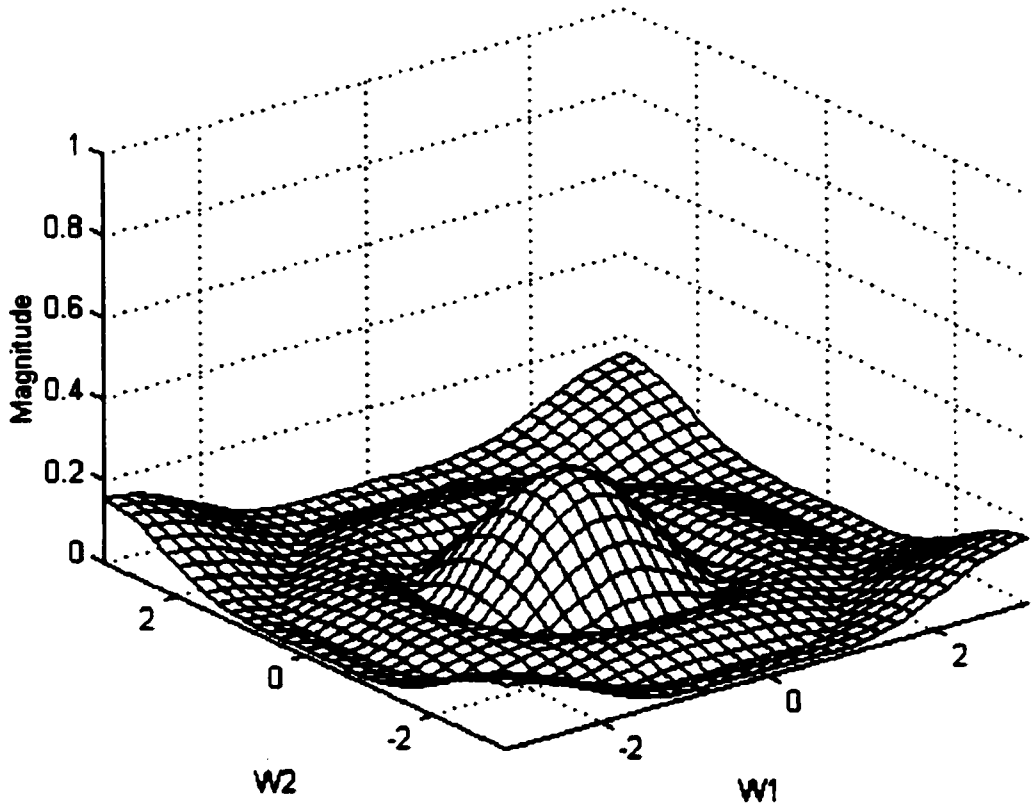

Figure 5.13 Magnitude response of the designed filter in example 5.3 at ω_3=1.3744

5.6 SUMMARY

In this chapter it has been shown that linear programming could be used to design different types of 3-D filters. It was used to design FIR filters with arbitrary magnitude response and the design of stable 3-D IIR filters with near linear phase. It was also use to design 3-D IIR filters with separable denominator and non-separable numerator. In the case of FIR and separable denominator filters symmetries were utilized to reduce the number of filter coefficients.

CHAPTER 6

DESIGN OF STABLE 3-D RECURSIVE DIGITAL FILTERS USING 3 -VARIABLE VERY STRICTLY HURWITZ POLYNOMIAL

6.1 Introduction

For many years researchers have relied on transformation of 2-variable passive analog network, using double bilinear transformation method to design 2-D recursive filters . This is done to avoid stability problem associated with IIR filters.

Goodman[40] showed that not all analog filters will yield stable digital filters upon the application of bilinear transformation.

Rajan et al.[41] showed that only a special class of analog filters can be transformed to a stable digital filters using bilinear transformation. This special class of 2-D analog filters should have Very Strictly Hurwitz Polynomial in their denominators. Two-variable VSHP's have been utilized for the design of stable 2-D digital filters by many researchers[42]-[44]

Recently, Liu and Bruton [45] have shown 3-D planar and beam filters can be designed by transformation of 3-variable passive RLC elements. Here we present a new method for generating 3-variable VSHP. In this method the properties of positive definite or positive semi-definite matrices and resistive matrix is utilized to generate the desirable 3-variable VSHPs. The presented methodology is an extension of the technique presented in [43] for 2-D VSHP. The application of the generated polynomials is shown through the design of 3-D recursive filters.

6.2 CHARACTERIZATION OF 3-D ANALOG AND RECURSIVE DIGITAL FILTERS.

A 3-D analog filter is characterized by its transfer function

$$H_a(s_1,s_2,s_3) = \frac{P_2(s_1,s_2,s_3)}{P_1(s_1,s_2,s_3)} \tag{6.1}$$

where

$$P_j(s_1,s_2,s_3) = \sum_{i_1=0}^{M_1}\sum_{i_2=0}^{M_2}\sum_{i_3=0}^{M_3} P(i_1,i_2,i_3)s_1^{i_1}s_2^{i_2}s_3^{i_3} \tag{6.2}$$

for j=1,2

The denominator of a stable filter must satisfy the following condition

$$P_1(s_1,s_2,s_3) = 0 \text{ for } \bigcap_{i=1}^{3} s_i \geq 0 \tag{6.3}$$

An equivalent 3-D digital filter transfer function can be obtained by the application of triple bilinear transformation of the variable s_1, s_2 and s_3 in eq. (6.1).

6.3 CONDITIONS FOR A 3- VARIABLE POLYNOMIAL TO BE A VSHP

We will start with a three variable polynomial given by

$$F(s_1,s_2,s_3) = s_3\, F_1(s_1,s_2) + F_2(s_1,s_2) \tag{6.4}$$

Where the degree of one of the variables say(s_3) is unity, and $F_1(s_1,s_2)$ and $F_2(s_1,s_2)$ are polynomials in two-variables s_1 and s_2. When s_3=0, $F(s_1,s_2,s_3)$ = $F_2(s_1,s_2)$ shall be VSHP. Similarly, when $s_3 \to \infty$, $F(s_1,s_2,s_3) \to F_2(s_1,s_2)$ which shall be also be a VSHP.
Therefore, in order that $F(s_1,s_2,s_3)$ shall be a VSHP, the polynomials $F_1(s_1,s_2)$ and $F_2(s_1,s_2)$ shall be 2-variables VSHPs.

The polynomial $F(s_1, s_2, s_3)$ will become zero when

$$s_3 = -\frac{F_2(s_1, s_2)}{F_1(s_1, s_2)} \tag{6.5}$$

In order that $F(s_1, s_2, s_3)$ is VSHP, its zeros shall lie strictly in the left-half of s_3-plane. This yield

$$\text{Re}\left(\frac{F_2(s_1, s_2)}{F_1(s_1, s_2)}\right) \geq 0, \text{ for } \text{Re}\, s_1 \geq 0, \text{ Re}\, s_2 \geq 0 \tag{6.6}$$

where Re represents the Real Part.

Writing

$$F_1(s_1, s_2) = (M_A + N_A) \tag{6.7}$$

and

$$F_2(s_1, s_2) = (M_B + N_B) \tag{6.8}$$

where

M_A = Even part of $F_1(s_1, s_2)$

N_A = Odd part of $F_1(s_1, s_2)$

M_B = Even part of $F_2(s_1, s_2)$

N_B = Odd part of $F_2(s_1, s_2)$

Eq(6.7) becomes equivalent to

$$M_A M_B - N_A N_B \succ 0 \text{ for } s_1 = j\omega_1 \text{ and } s_2 = j\omega_2 \tag{6.9}$$

With this condition, it is readily seen that $F(s_1, s_2, s_3)$ will be a three-variables VSHP.

From the above discussion, we have proved the following theorem.

Theorem 1 A three-variable polynomial of type given in eq.(6.4) , where one of the variables is of degree unity, will be a VSHP, when the following conditions are satisfied:

(i) $F_1(s_1, s_2)$ and $F_2(s_1, s_2)$ are two-variables VSHPs.

(ii) Real part of $\dfrac{F_2(s_1, s_2)}{F_1(s_1, s_2)}$ is positive in the poly-domain $\mathrm{Re}\ s_1 \geq 0$ and $s_2 \geq 0$.

the degree of the variable s_3 is unity and this may appear restrictive. this is not the case, because a higher degree in s_3 can be obtained be repeating application of VSHP transformations[51].

6.4. GENERATION OF 3-VARIABLE VSHP

In this method 2-variables VSHP is generated by using the following relationship

$$F_1(s_1, s_2) = \det\left[\sum_{i=1}^{2} A_i \Gamma_i A_i^T s_i + D \Delta D^T + G\right] \tag{6.10}$$

Where

A_i's and D are the upper triangular matrices with unity elements on their diagonal,

Γ_i's and Δ are diagonal matrices with non-negative elements, and G is a skew-symmetric matrix. In view of VSHP transformations, we can attempt to generate 2-variable polynomial in which the degree of each variable is unity. In such a case, these matrices are as follows:

$$A_i = \begin{bmatrix} 1 & a_i \\ 0 & 1 \end{bmatrix} \quad D = \begin{bmatrix} 1 & d \\ 0 & 1 \end{bmatrix} \quad \Gamma = \begin{bmatrix} \alpha_{i1}^2 & 0 \\ 0 & \alpha_{i2}^2 \end{bmatrix} \quad \Delta = \begin{bmatrix} \beta_1^2 & 0 \\ 0 & \beta_2^2 \end{bmatrix} \text{ and } G = \begin{bmatrix} 0 & g \\ -g & 0 \end{bmatrix} \tag{6.11}$$

A SHP polynomial of 2-variables will be obtained by the substitution of the above matrices in eq.(6.10). This yields

$$F_1(s_1,s_2)=\sum_{i=1}^{2}\alpha_{i1}^2\,\alpha_{i2}^2\,s_i+\sum_{i=1}^{2}\sum_{j=1}^{2}\alpha_{i1}^2\alpha_{j2}^2+\alpha_{j1}^2\alpha_{i2}^2+\alpha_{j2}^2\alpha_{i2}^2(a_i-a_j)^2 s_i s_j + \sum_{i=1}^{2}\left[\alpha_{i1}^2\beta_2^2+\alpha_{i2}^2\beta_1^2+\alpha_{i2}^2\beta_2^2(a_i-d)^2\right]s_i+g^2+\beta_1^2\beta_2^2 \tag{6.12}$$

Very Strictly Hurwitz Polynomial is obtained by substituting $\alpha_{i1}=0$ in eq.(6.12) yielding

$$F_1(s_1,s_2)=\sum_{i=1}^{2}\sum_{j\neq i}^{2}\alpha_{j2}^2\alpha_{i2}^2(a_i-a_j)^2 s_i s_j + \sum_{i=1}^{2}\left[\alpha_{i2}^2\beta_{j1}^2+\alpha_{i2}^2\beta_{j2}^2(a_i-d)^2\right]s_i+g^2+\beta_1^2\beta_2^2 \tag{6.13}$$

As can be seen from eq.(6.13), terms s_i^2 are not present in VSHP as opposed to SHP. In order that $F_1(s_1,s_2)$ is a 2-variable VSHP, it is required that

$$a_1 \neq a_2 \tag{6.14}$$

Alternatively, one can start with a polynomial

$$Q(s_1,s_2)=a_{11}\,s_1s_2+a_{10}\,s_1+a_{01}\,s_2+a_{00} \tag{6.15}$$

The polynomial $Q(s_1,s_2)$ will be a 2-variable VSHP, when

$$a_{11}>0,\ a_{10}>0,\ a_{01}>0 \text{ and } a_{00}>0 \tag{6.16}$$

are satisfied. In fact, eq.(6.15) and eq. (6.16) will be the same, when

$$a_{11} = \alpha_{12}^2 \alpha_{22}^2 (a_1 - a_2)^2$$

$$a_{10} = \left[\alpha_{12}^2 \beta_1^2 + \alpha_{12}^2 \beta_2^2 (a_1 - d)^2\right]$$

$$a_{01} = \left[\alpha_{22}^2 \beta_1^2 + \alpha_{22}^2 \beta_2^2 (a_2 - d)^2\right] \tag{6.17}$$

$$a_{00} = g^2 + \beta_1^2 \beta_2^2$$

By making $F_1(s_1, s_2)$ to be $B(s_1, s_2)$ or $Q(s_1, s_2)$, it is required to generate $F_2(s_1, s_2)$ such the condition of theorem 1 is satisfied

Method I. The polynomial $F_2(s_1, s_2)$ can be generated as enunciated earlier and let it be given by

$$F_2(s_1, s_2) = b_{11}\, s_1 s_2 + b_{10}\, s_1 + b_{01}\, s_2 + b_{00} \tag{6.18}$$

and this will be a 2-variable VSHP provided that

$$b_{11} > 0,\ b_{10} > 0,\ b_{01} > 0 \text{ and } b_{11} > 0 \tag{6.19}$$

Consider the rational function

$$G_1(s) = \frac{a_{11}\, s_1\, s_2 + a_{10}\, s_1 + a_{01}\, s_2 + a_{00}}{b_{11}\, s_1\, s_2 + b_{10}\, s_1 + b_{01}\, s_2 + b_{00}} \tag{6.20}$$

$$M_A = (a_{11} s_1 s_2 + a_{00}) \tag{6.21}$$

$$N_A = (a_{10} s_1 + a_{01} s_2) \tag{6.22}$$

$$M_B = (b_{11} s_1 s_2 + b_{00}) \tag{6.23}$$

$$N_B = (b_{11} s_1 + b_{00} s_2) \tag{6.24}$$

This yields

$$M_A M_B - N_A N_B\big|_{S_1 = j\omega_1,\ S_2 = j\omega_2} =$$

$$a_{11}b_{11}\omega_1^2\omega_2^2 + \omega_1\omega_2(a_{10}b_{01} + a_{01}b_{10} - a_{00}b_{11} - a_{11}b_{00}) + a_{10}b_{10}\omega_1^2 + a_{01}b_{01}\omega_2^2 + a_{00}b_{00} > 0 \tag{6.25}$$

After some algebraic manipulations, it can be shown that eq(25) is equivalent to the condition

$$[4(a_{01}b_{01}a_{01}b_{01} + a_{00}b_{00}a_{11}b_{11}) - (a_{10}b_{01} + a_{01}b_{10} - a_{00}b_{11} - a_{11}b_{00})^2]^2 - 64a_{01}b_{01}a_{11}b_{11}a_{00}b_{00}a_{10}b_{10} < 0 \tag{6.26}$$

holds.

It is also observed that eq.(26) is satisfied under a sufficient condition

$$a_{10}b_{01} + a_{01}b_{10} = a_{00}b_{11} + a_{11}b_{00} \tag{6.27}$$

When we generate $F_1(s_1, s_2)$ and $F_2(s_1, s_2)$ care must be taken to see that condition (6.25) or (6.26) is satisfied. This enables us to generate the polynomial $P_1(s_1, s_2, s_3)$ as

$$s_3 F_1(s_1, s_2) + F_2(s_1, s_2) \tag{6.28}$$

$$s_3 F_2(s_1, s_2) + F_1(s_1, s_2) \tag{6.29}$$

Method II : As an alternative, we can generate a large class of functions which satisfy the condition (6.6). It is already shown that a two-variable VSHP $F_1(s_1, s_2)$ can be generated.

Consider

$$G_2(s_1,s_2)=\frac{K_1F_1(s_1,s_2)+K_2\dfrac{\partial F_1(s_1,s_2)}{\partial s_1}+K_3\dfrac{\partial F_1(s_1,s_2)}{\partial s_2}}{F_1(s_1,s_2)} \tag{6.30}$$

Obviously, the real part of $G_2(s_1,s_2)$ is given by

$$\mathrm{Re}(G_2(j\omega_1,j\omega_2))=K_1+\frac{K_2(a_{00}a_{10}+a_{01}a_{11}\omega_1^2)+K_3(a_{00}a_{01}+a_{10}a_{11}\omega_2^2)}{(a_{00}-a_{11}\omega_1\omega_2)^2+(a_{10}\omega_1+a_{01}\omega_2)^2} \tag{6.31}$$

which is always positive whenever $K_1>0$, $K_2>0$ and $K_3>0$. In such a case, the numerator of $G_2(s_1,s_2)$ {which can be called $F_2(s_1,s_2)$ } is also a 2-variable VSHP. It can be noted that either K_2 or K_3 can be made equal to zero but not both. This is because, if $K_2=K_3=0$, $G_2(s_1,s_2)$ will equal to K_1. There is a possibility that K_2 and K_3 can be made negative, provided the value of K_1 is made sufficiently high so that $\mathrm{Re}(G_2(s_1,s_2))$ remains positive. As can be seen, a very large class of such polynomials can be generated. A three variable VSHP can be obtained by using (6.28) and (6.29).

6.5. DESIGN TECHNIQUE

In the proposed design technique, the derived 3-variables VSHP is assigned to the denominator of analog transfer function as follows:

$$H(s_1,s_2,s_3)=\frac{N(s_1,s_2,s_3)}{D(s_1,s_2,s_3)} \tag{6.32}$$

The digital equivalent is obtained by using triple bilinear transformation [51]. The mean square error between the ideal magnitude response $|H_I(\omega_1,\omega_2,\omega_3)|$ and the designed magnitude

response $\left|H_D(\omega_1,\omega_2,\omega_3)\right|$ is used as an objective function for optimization:

$$E_{Mag}(\omega_1,\omega_2,\omega_3,\phi)=\left|H_I(\omega_1,\omega_2,\omega_3)\right|-\left|H_D(\omega_1,\omega_2,\omega_3,\phi)\right| \tag{6.33}$$

and

$$E_I^2(\omega_1,\omega_2,\omega_3,\phi)=\sum_{\omega_1,\omega_2,\omega_3\,\&\,I_{PS}}\sum\sum E_{Mag}^2(\omega_1,\omega_2,\omega_3,\phi) \tag{6.34}$$

where ϕ is the coefficient vector calculated by minimizing eq(33), and I_{PS} is the set of all frequencies in the passband and stopband. Minimization of this objective function is carried out using nonlinear Matlab. optimization technique to find the vector ϕ.

Design Example

A three dimensional digital filter with the following amplitude specifications is designed by using VSHP polynomial:

$$\left|H_I(\omega_1,\omega_2,\omega_3)\right|=\begin{cases}1 \text{ for } 0\le\sqrt{\omega_1^2+\omega_2^2+\omega_3^2}\le 1\\ 0 \text{ for } 2\le\sqrt{\omega_1^2+\omega_2^2+\omega_3^2}\le\pi\end{cases}$$

For this purpose three 3-variable VSHP of the form (28) or (29) are cascaded and the digital form is obtained by using triple bilinear transformation as follows

$$D_1(s_1,s_2,s_3)=s_3\,F_1(s_1,s_2)+F_2(s_1,s_2)$$

$$D_2(s_1,s_2,s_3)=s_2\,F_3(s_2,s_3)+F_4(s_2,s_3)$$

$$D_3(s_1,s_2,s_3)=s_1\,F_5(s_2,s_3)+F_6(s_2,s_3)$$

$$D(z_1,z_2,z_3) = D_1(s_1,s_2,s_3) \times D_2(s_1,s_2,s_3) \times D_3(s_1,s_2,s_3)\Big|_{S_i \to \frac{2}{T}\frac{1-z_i^{-1}}{1+z_i^{-1}}} \qquad i = 1, 2, 3$$

The designed frequency is

$$H_D(z_1,z_2,z_3) = \frac{\sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3} a(i,j,k) z_1^{-i} z_2^{-j} z_3^{-k}}{D(z_1,z_2,z_3)}$$

The coefficients of the designed filter obtained are shown in table 1, figure 1, 2, 3 and 4 show the magnitude response of the designed filter at different values of ω_3.

6.6 CONCLUSION

In this chapter, methods are presented for generation of 3-variable VSHP. It is shown how these VSHP's can be used to design 3-D stable recursive digital filters. example is given to illustrate the usefulness of the propped approach.

Table 6.1 Coefficients of the filter of example 6.1

$A_{0,0,0}$ =0.3820	$A_{0,1,0}$ =-0.0331	$A_{0,2,0}$ = -0.1629	$A_{0,3,0}$ =0.1532
$A_{0,0,1}$ =0.1893	$A_{0,1,1}$ =0.0258	$A_{0,2,1}$ = -0.0212	$A_{0,0,1}$ =0.0359
$A_{0,0,2}$ = -0.1344	$A_{0,1,2}$ =-0.0427	$A_{0,2,2}$ =0.1600	$A_{0,3,2}$ =0.0210
$A_{0,0,3}$ =-0.0935	$A_{0,1,3}$ =0.0558	$A_{0,2,3}$ = -0.0288	$A_{0,3,3}$ =0.0274
$A_{1,0,0}$ =0.2109	$A_{1,1,0}$ = 0.0639	$A_{1,2,0}$ = -0.1031	$A_{1,3,0}$ =0.0702
$A_{1,0,1}$ =0.0516	$A_{1,1,1}$ =0.0485	$A_{1,2,1}$ = -0.0909	$A_{1,3,1}$ =0.1723
$A_{1,0,2}$ = -0.1686	$A_{1,1,2}$ =-0.1105	$A_{1,2,2}$ = 0.0824	$A_{1,3,2}$ -0.0087
$A_{1,0,3}$ =-0.1098	$A_{1,1,3}$ =-0.0386	$A_{1,2,3}$ = -0.0526	$A_{1,3,3}$ = 0.0296
$A_{2,0,0}$ =-0.1074	$A_{2,1,0}$ = 0.0940	$A_{2,2,0}$ =0.1408	$A_{2,3,0}$ =0.0135
$A_{2,0,1}$ =0.0501	$A_{2,1,1}$ = -0.0460	$A_{2,2,1}$ = 0.0539	$A_{2,3,1}$ = 0.0023
$A_{2,0,2}$ = -0.1125	$A_{2,1,2}$ = 0.0252	$A_{2,2,2}$ =0.0579	$A_{2,3,2}$ =0.0048
$A_{2,0,3}$ = -0.0433	$A_{2,1,3}$ = 0.0161	$A_{2,2,3}$ =-0.0866	$A_{2,3,3}$ = -0.0132
$A_{3,0,0}$ =-0.1360	$A_{3,1,0}$ =0.0021	$A_{3,2,0}$ =0.0452	$A_{3,3,0}$ =0.0148
$A_{3,0,1}$ = -0.1073	$A_{3,1,1}$ = -0.0482	$A_{3,2,1}$ =-0.0120	$A_{3,3,1}$ =0.0195
$A_{3,0,2}$ =0.0315	$A_{3,1,2}$ =0.0498	$A_{3,2,2}$ =-0.0405	$A_{3,3,2}$ =0.0071
$A_{3,0,3}$ = -0.0321	$A_{3,1,3}$ =0.0228	$A_{3,2,3}$ =0.0187	$A_{3,3,3}$ =-0.0187

Table 6.1 continued

	i=1	*i=2*	*i=3*	*i=4*	*i=5*	*i=6*
a_i	1.0029	0.9990	1.0435	0.9966	0.9316	0.9979
α_{i1}	1.000	1.0000	1.0000	1.0000	1.0000	1.0000
α_{i2}	0.9343	0.9916	0.8996	0.9751	0.7714	1.0443
b_i	0.9229	0.8429	1.1288	0.8746	1.1086	1.0343
β_{i1}	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
β_{i2}	0.9609	0.9909	0.9866	0.9811	0.9728	0.9651
c_i	1.0380	1.0809	0.9237	1.0659	0.9306	0.9825
γ_{i1}	0.9339	0.9542	0.9096	0.9378	0.9353	1.0098
γ_{i2}	1.0425	1.0235	1.0747	1.0261	1.0237	0.9911
g_i	1.0358	0.9997	1.0422	1.0075	1.0813	0.9969

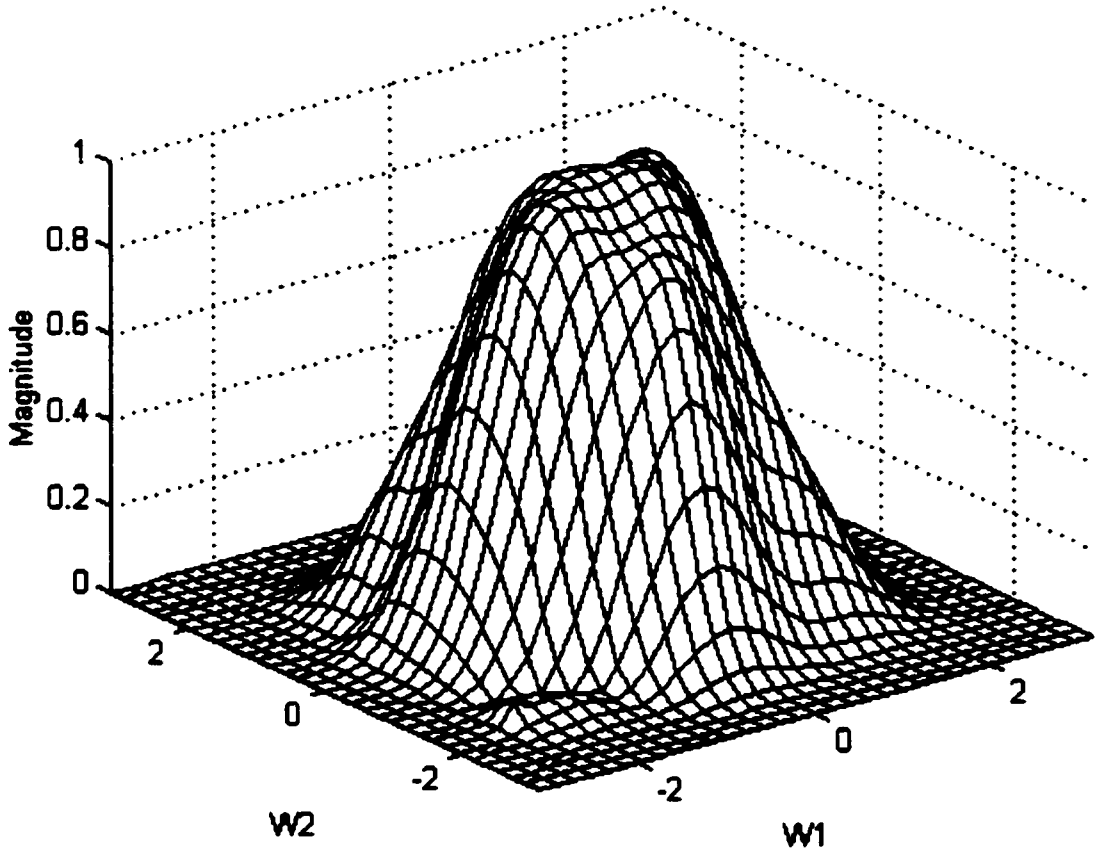

Fig 6.1 Magnitude response of the designed filter at ω_3=0

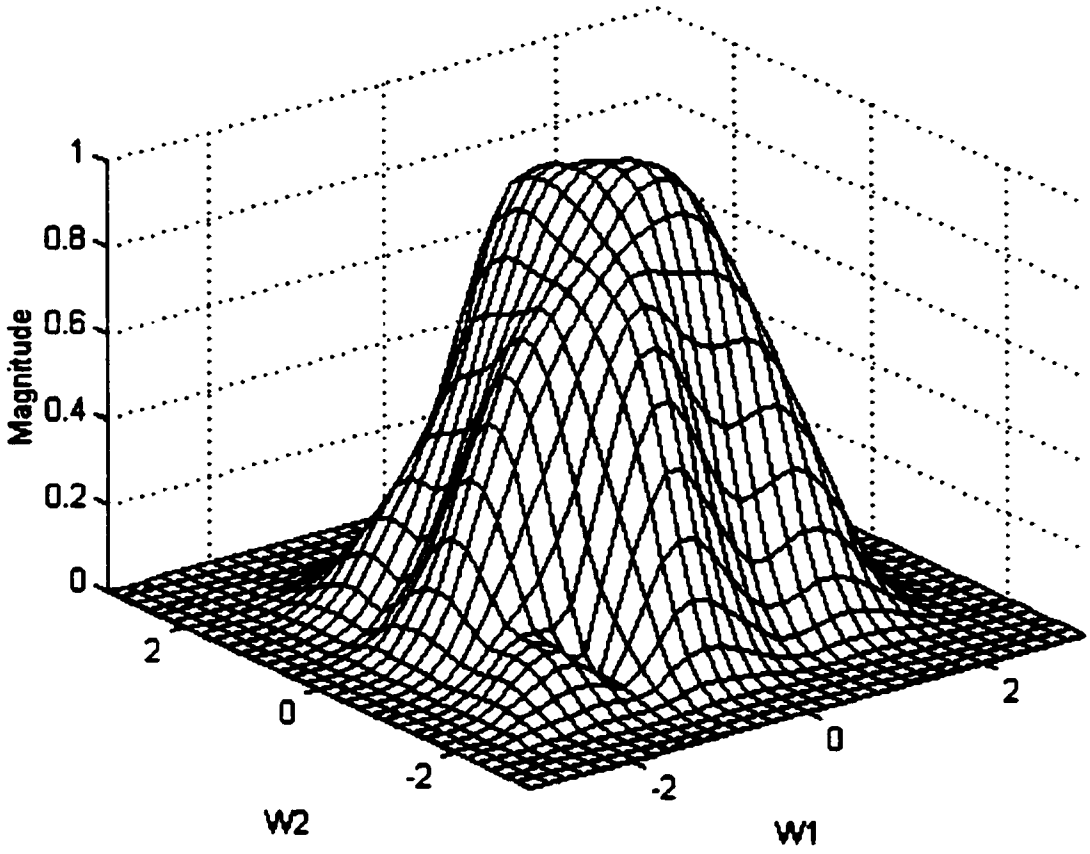

Fig 6.2 Magnitude response of the designed filter at $\omega_3 = 0.3927$

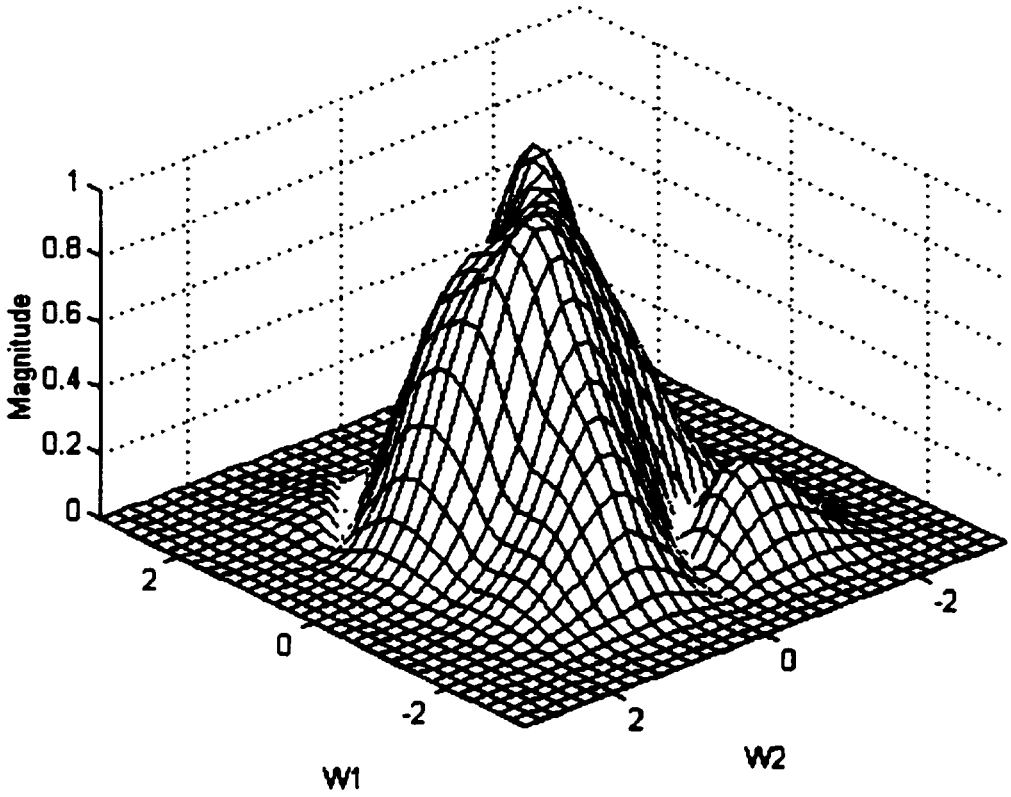

Fig 6.3 Magnitude response of the designed filter at ω_3 =0.9817

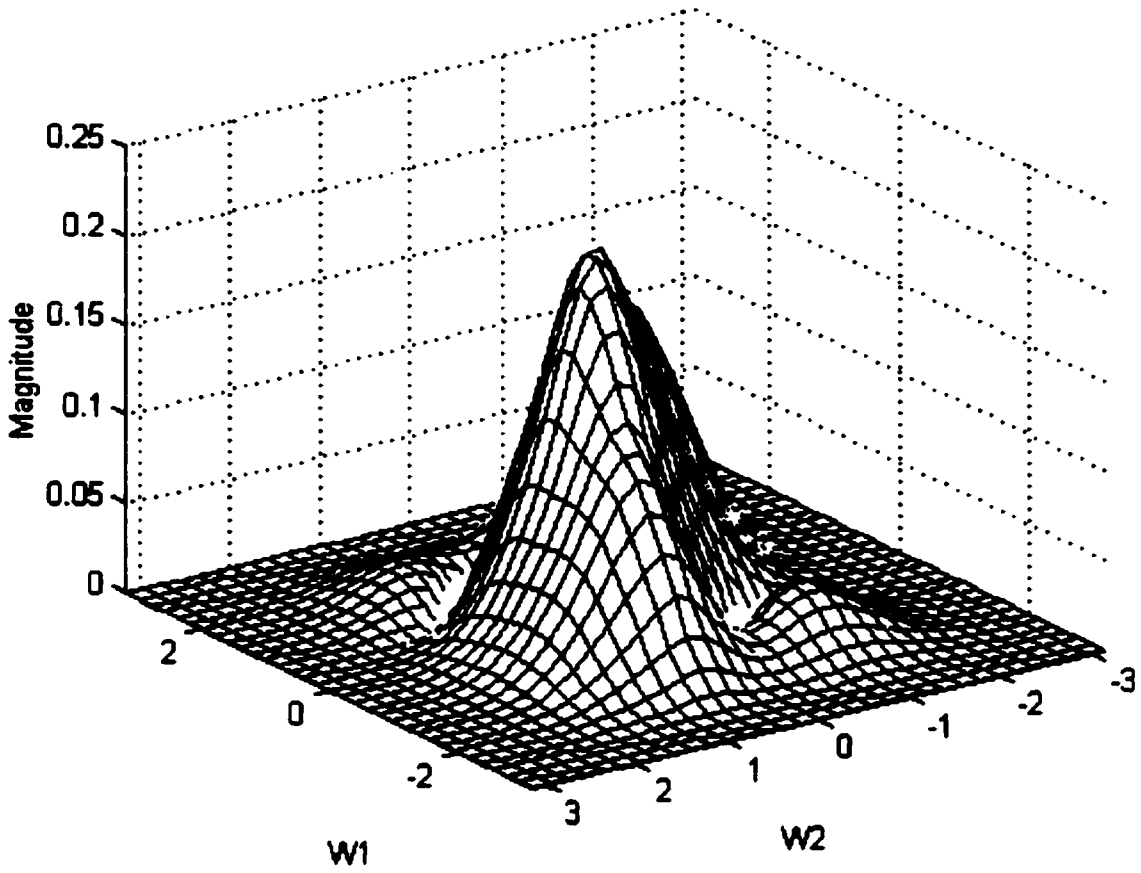

Fig 6.4 Magnitude response of the designed filter at $\omega_3 = 1.9635$

CHAPTER SEVEN

SUMMARY AND CONCLUSION

This thesis is concerned with the design techniques for linear-phase no-recursive filters and near linear-phase recursive filters in three dimensions. It should be noted that the techniques used here lend themselves to the flexibility of designing virtually any type of filter besides the standard filters such as lowpass, highpass or bandpass filters. The first techniques introduced , used for the design of FIR filters, is a Fourier Series Expansion design method . Due to the slow convergence of Fourier series, filters designed using this method suffered from Gibb's oscillation. In chapter three we developed an algorithm that utilizes the Fast Fourier Transform (FFT) in the design of 3-D FIR filters. To reduce Gibb's oscillations window functions were utilized. Although this method is readily extended to 3-D and enjoy short design time, results in filters with an enormous number of coefficients. Different symmetries were proposed to significantly reduce the number of coefficients and the region of frequency region. Good approximation of spherical spherically symmetric, lowpass and highpass can be achieved using the proposed method. In chapter 4 we extended Shank's method to design a stable, near-linear phase 3-D digital filters. To ensure the stability of the designed filter we started with a decaying impulse response while the linear phase is achieved by shifting the impulse response. In this method Gauss-Jordan's elimination method is used to solve the set of linear equations. In Chapter 5 we unitized linear programming approach to design 3-D FIR filters. The

number of coefficients were substantially reduced by the use of different symmetries. The linear programming approach was also used to design a stable 3-D IIR filters. This techniques was use to approximate 3-D IIR transfer function by approximating simultaneously linear phase and arbitrary magnitude response. This technique sets u a linear programming problem which in term of filter coefficients and the desired constant group delay(linear phase),which is then solved interactively for the best approximation to the given specifications. Since the approximation is performed using linear programming, the constraints on filter coefficients to produce a stable filter are required to be linear in form. It is known that many useful 3-D filters can be designed using a 3-D transfer functions with a separable denominator. This choice of transfer function offers a great reduction in number of filter coefficients, hence saving computer time and implementation cost. Efficient design method was outlined. In this method the numerator and denominator of 3-D transfer function is calculated separately. In the first phase the optimization, 3 all poles 1-D filters are designed to satisfy the specifications along ω_1, ω_2 and ω_3 axis. Then, by cascading this rectangular cutoff boundary with 3-D FIR filter, the desired shape of the cutoff boundary is obtained. By using cubic, diagonal and 16-hydral symmetries the number of filter coefficients and the frequency region are reduced by more than 90% of the total number. Based upon the properties of positive definite matrices, a method for generating three variable VSHP of arbitrary order and their use in the design of a stable 3-D recursive filters was implemented in chapter 6. Example to demonstrate its use in the design utilizing non-linear optimization is presented. It is worth mentioning that stability of filters designed using this method is guaranteed.

Designing stable 3-D filters is a difficult task, since a general 3-D function cannot be factored into lower order polynomials. Therefore not all 1-D or even 2-D design techniques can be extended to 3-D. Furthermore the stability checking of 3-D recursive filters becomes cumbersome. Development of many stability criteria and 3-D digital filters realization would be a very important research area.

REFERENCES

[1] G.T. Herman, "On the Nois in Images Produced by Computed Tomography," *Computer graphics and image Processing*, Vol. 12, pp.271-285, 1980.

[2] L. Axel, P.H. Arger and R.A. Zimmerman, "Application of Computer Tomography to Diagnostic Radiology," *proceedings of IEEE*, Vol. 17, No. 5, pp. 293-297, March 1983.

[3] J.P. Burg, "Three-Dimensional Filtering with an Array of Seismometers," *geophysics*, Vol. 71, No. 3, pp.293-297, March 1983.

[4] D. T. Kuan. "Three dimensional vision system for object Recognition'' *proc. of SPIE, Intelligent Robots*, Vol. 449, pp 366-372

[5] L. T. Bruton and N. R. Bartley, "Three-Dimensional Image Processing Using the Concept of Network Resonance," IEEE Trans. on Circuits and Systems, Vol. CAS-32, No.7, pp 664-672, July 1985.

[6] Zhang, Y. and Bruton, L.T., "Applications of 3D LCR networks in the design of 3D recursive filters for processing image sequences," *IEEE Trans. on Circuits and Systems for Video Technology*, Vol. 4, No. 4, pp. 369-382, August 1994.

[7] M. Zervakis,A. Venetstanopoulos,"Design of Three-Dimensional Digital Filters Using Two-Dimensional Rotated Filters,"IEEE Trans. on Circuits and Systems, Vol. CAS-34, No.12, Dec. 1987.

[8] M. A. Sid-Ahmed, "Image Processing Theory Algorithms and Applications," McGraw Hill, New York, 1994.

[9] John K. Pitas and Anastasion N. Venetsanopoulos, "The use of Symmetries in the Design of multidimensional Digital filters," *IEEE Trans. Circuits Syst.* Vol. CAS-33, pp 863-873, Sept.986

[10] J. H. Lodge and M. Fahmy, "Symmetry in Two-dimensional Rectangularly Sampled Digital Filters," *IEEE Trans. Acoust.,Speech. Signal Processing*, Vol. ASSP-29, No.4, pp. 794- 805, Aug.1981.

[11] B.D.O. Anderson & E.I. Jury, "Stability of Multi-dimensional Digital Filters," *IEEE Trans. Circuit & Syst.* CAS-21, No. 2, pp. 300-304, March 1974.

[12] J.H. Justice & J.L. Shanks, "Stability Criterion for N-Dimensional Digital Filters," *Trans. Automatic Control*, AC-18, No.3, pp. 284-286, June 1973.

[13] M.G. Strintziz, "Test of Stability of Multidimensional Filters," *IEEE Trans. Circuits and Syst. Vol. 25*, pp.657-675, May 1977.

[14] P. Agathoklis and L. T. Bruton, "Practical-BIBO Stability on N-dimensional Discrete Systems," Proc. Inst. Elec. Eng., Vol. 130, pt. G, No. 6, pp. 236-242, Dec. 1983.

[15] M.N.S Swamy,Leonid M. Roytman and Eugene I. Plotkin, "On Stability Properties of Three and Higher Dimensional Linear Shift-Invariant Digital Filters," *IEEE Trans. on Circuit and Systems*, Vol Cas-3, No.9, Sept 1985.

[16] R. King, M. Ahmadi, R. Nagib, A. Kwabwe, and M. Azimi Sadjadi, " Digital Filtering in one and two Dimensions, Design and Applications," Plenum Press,1989.

[17] J. H. McClellan, "The Design of Two-Dimensional Digital Filters by Transformation," Proc. 7th Annual Princeton Conf. Information Science and Systems, pp. 247-251, 1973

[18] C.K. Yan and A.N. Venetsanopoulos, "Transformation Design of Three-Dimensional FIR Filters," *Proceedings of ELECTROCOM 85*, Toronto, Canada, Oct. 1985.

[19] Michael E. Zervakis, "Design of 3-D Digital Filters using Transformations ," Msc Thesis, University of Toronto, Toronto, Canada.

[20] L. J. Karam, "On the Design of Multidimensional FIR Filters by Transformation," *IEEE International Conference on Acoustics, Speech, and Signal Processing* , April 1997.

[21] A. Fetweis, "Symmetry Requirements for Multidimensional Digital Filters," *IEEE Circuit Theory and Applications*, vol 5, 343-353 (1977).

[22] L.R. Rabiner and J.V. Hu, "Application of Linear Programming Technique to the Design of Finite-Duration Impulse Response Digital Filters," Proc. Digital Filtering, Imperial College, London, England, 1971.

[23] J.V. Hu,"Frequency Domain Design of Two-Dimensional Finite impulse Response Digital Filters," Proc. Two-Dimensional Digital Signal Processing Conf. University. Missouri, Colmbia, Oct. 1971.

[24] T. Higuchi, M. Ohki and M. Kawamata, " Optimal Design of Three-Dimensional FIR Digital Filters- Design Method by Linear Programming and Reduction of Computation by Exploiting Symmetries," Trans. I.E.I.C.E., J70-A, pp. 1042-1050. July 1987.

[25] B. Nowrouzian, M. Ahmadi, r. A. King, " On the Space Domain Design Techniques of N-Dimensional Recursive Digital Filters," *IEEE International Conference, Speech and Signal processing Record*, pp. 531-534. May 1977.

[26] K. Hirano, M. Sakane and M. Z. Mulk. "Design of Three-Dimensional Recursive Digital Filters," *IEEE Trans. Circuits & Syst.*, CAS-31, pp. 550-561 June 1984.

[27] I. El-Feghi, M. Ahmadi, M. A. Sid-Ahmed, " Design of 3-D Recursive Digital Filters With Separable Denominator and Non-separable Numerator," Accepted in WAC 2000, Manu, Hawaii. June, 2000.

[28] M. Zervakis nad A. Venetsanopoulos, "Design of Three-Dimensional Digital Filters using Two-Dimensional Rotated Filters," *IEEE Trans. On Circuits and Systems*. Vol. CAS-34, No.12. Dec.1987.

[29] Takao Hinmoto and Ken-ichi Harada, "Design of 3-D Separable-Denominator Digital Filters Using Minimal Decomposition and Balanced Realization," *Electronics and Communications in Japan*, Part 3, Vol. 77, No. 10, 1994.

[30] T. Hintamoto, T. Hamanaka and S. maekawa , "Design of Three-Dimensional Separable-Denominator Digital Filters via Singular Value Decomposition ," Trans. I.E.I.C.E., J70-A, pp.785-790 , May 1987.

[31] Kalman, R. E, "Design of self-optimizing control system," *Trans. ASME, 80*, pp. 468-478.1958.

[32] Steiglitz, K. and McBride, L.E, "A Technique for the Identification of Linear Systems," *IEEE Trans. Automatic Control*, AC-10, pp. 461-484. 1965.

[33] Shank, J. L, "Recursion filters for Digital Processing," *Geophysic*, 32, pp.33-51. 1967.

[34] Bordner, G. W. "Time Domain Design of Stable Recursive Digital Filters," Ph.D. Dissertation, Dept. of Electrical Engineering, State university, New York, Buffalo. 1974.

[35] Bertran, M. S, " Approximation of Digital Filters in One and Two Dimensions," *IEEE Trans. Acoust. Speech, Signal processing*, ASSP-23, pp. 438-443.

[36] Yongbing Wan and M. Fahmy, "Design of N-D Digital Filters with Finite World Length," *IEEE Tran. Circuits and Syst.* , Vol. 36, No 3, March 1989.

[37] A.T. Chottera and G.A. Jullien, "Design of two-Dimensional Recursive digital Filters Using Linear Programming ," IEE Trans. On Circuits and Systems, Vol. CAS-29, No. 12, Dec. 1982

[38] L.R. Rabiner, N.Y. Graham, and M.D. Helms, " Linear Programming Design of IIR Digital Filters with Arbitrary Magnitude Function," *IEEE Tran. Acoust., Speech, Signal Processing* ASSP-22, pp. 117-123, 1974.

[39] I. El-Feghi, M.A. Sid-Ahmed, and M. Ahmadi, 'Design of 3-D Recursive Digital Filters using Linear Programming," 5th ISSPA '99', Brisbane, Australia, pp. 973-976, Aug. 1999

[40] D. Goodman, "Some Difficulties With Double Bilinear Transformation in 2-D digital Filter design Transfer Function," *IEEE Trans, on Circuits and Systems*, Vol. CAS-25, No. 6,pp.340-343, June 1978.

[41] P.K. Rajan.H. C. Reddy, M. N. S. Swamy, and V. Ramchandran, "Generation of Two-Dimensional Digital Function Without Nonessential Singularities of the Second kind," *IEEE Trans. Acoust. Speech and Signal processing* Vol. ASSP-28, No. 2 pp. 216-223, April 1980.

[42] V.Ramachandran and M. Ahmadi, "Design of 2-D stable and Recursive Digital Filters Using Properties of Derivatives of Even or Odd parts of Hurwitz polynomials," *Journal of franklin Institute*, Vol. 315, No. 4, pp.259-267, 1983.

[43] M Ahmadi and V.Ramachandran, A New Method for Generating 2-Variables VSHP and its Application in the Design of 2-D Recursive Filters with Prescribed Magnitude and Constant Group-Delay Response," *Proc. IEE, Part G, Electronics Circuits and Systems*, Vol.131, pp. 151-155, 1984.

[44] M. Ahmadi, A. Mazinani, V.Ramachandran and M. Shridhar, A computationally Efficient Technique for Deriving 2-Variables VSHP," *Jouranl of Computers and Electrical Engineering*, Vol.21, No.2,pp.101-106,1995.

[45] Q.Liu and L.T.Bruton," Design of 3-D Planar and Beam Recursive Digital Filters Using Spectral Transformation," *IEEE Transaction on Circuits and Systems*, Vol. CAS-36, pp.365-374,1989.

[46] I. El-Feghi, M. Ahmadi, W. C. Miller ,V. Ramachandran, M. A. Sid-Ahmid, "A Method for Generation of 3-Variables VSHP and its application in Design of Stable 3-Dimensional Recursive Digital Filters," Accepted in WAC 2000, Manu, Hawaii. June, 2000.

[47] Abramowitz and Stegun, "HandBook of Mathematical Functions," Dover publication Inc., New york, 1964

[48] P.K. Rajan, "A Study on the properties of Multi-Dimensional Fourier Transforms," *IEEE Trans. circuits Syst.*, Vol . CAS-31, Aug. 1984.

[49] D.G. Luenberger, "Linear and Non-Linear programming," Second Edition, Addison-Wesley, Reading,1984.

[50] Appanna T. Chottera, "Recursive Digital Filters: Design and Applications to Image Processing," Ph.d Dissertation, University of Windsor, Windsor, Canada. 1979.

[51] V.Ramachandran and C.S. Gargour, "Generation of Very Strictly Hurwitz Polynomials and Applications to 2-D Filter Design, Multidimensional System," *Signal Processing and Modeling Techniques*, Vol.69, Academic Press Inc.,1995, pp. 211-254.

APPENDIX A

DESIGN OF 3-D FIR FILTERS USING NUMERICAL INTEGRATION COMPUTER PROGRAMS

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
import java.io.*;
import com.sun.java.swing.*;
import borland.jbcl.layout.*;
import borland.jbcl.control.*;
public class Main extends JApplet {
  boolean isStandalone = false;
  private Filter F;
  private Frame1 FR;
  private Dsp_Class C;
  private File_Chooser FCH;

  int F_Type=1, I, J, K, num1, num2, num3;
  double Impulse[][][];
  RandomAccessFile I_File, F_File;
  XYLayout xYLayout1 = new XYLayout();
  Label Filter_ORder = new Label();
  TextField N1 = new TextField();
  TextField N2 = new TextField();
  TextField N3 = new TextField();
  Label X1 = new Label();
  Label X2 = new Label();
  CheckboxGroup Filter_Type_Group = new CheckboxGroup();
  Label Cut_Off_Label = new Label();
  TextField Cut_Off_Value = new TextField();
  Label I_R_FLabel = new Label();
  Label F_R_FLabel = new Label();
  TextField F_R_File_Text = new TextField();
  Button New_button = new Button();
  Button Clear_button = new Button();
  Button Cancell_button = new Button();
  Label Filter_type = new Label();
  Checkbox High_Pass_check = new Checkbox();
  Checkbox Low_Pass_check = new Checkbox();
  TextField I_R_File_Text = new TextField();
  Label label2 = new Label();
  Label label1 = new Label();
  Label label3 = new Label();
  GroupBox Status = new GroupBox();
  Button Diplay_button = new Button();
  JFrame frame1=new JFrame();
```

```
 JProgressBar jProgressBar1 = new JProgressBar();
 //Get a parameter value
 public String getParameter(String key, String def) {
  return isStandalone ? System.getProperty(key, def) :
    (getParameter(key) != null ? getParameter(key) : def);
 }
 //Construct the applet
 public Main() {
 }
//Initialize the applet

 public void init() {
  try {
  jbInit();
  }
  catch (Exception e) {
  e.printStackTrace();
  }
 }
com.sun.java.swing.plaf.windows.WindowsLookAndFeel());
 private void jbInit() throws Exception {
  Low_Pass_check.setFont(new Font("Dialog", 1, 15));
  xYLayout1.setHeight(648);
  Filter_ORder.setFont(new Font("Dialog", 1, 17));
  Filter_ORder.setText("Filter Order  :");
  N1.setBackground(Color.white);
  N2.setBackground(Color.white);
  N3.setBackground(Color.white);
  N3.addKeyListener(new java.awt.event.KeyAdapter() {
public void keyReleased(KeyEvent e) {
      N3_keyReleased(e);
    }
  });
//-----------------------------
  X1.setFont(new Font("Dialog", 1, 15));
  X1.setText("X");
  X2.setFont(new Font("Dialog", 1, 15));
  X2.setText("X");
  Cut_Off_Label.setFont(new Font("Dialog", 1, 15));
  Cut_Off_Label.setText("Cut Off Frequency :");
  Cut_Off_Value.setBackground(Color.white);
  Cut_Off_Value.addKeyListener(new java.awt.event.KeyAdapter() {
    public void keyPressed(KeyEvent e) {
```

```
      Cut_Off_Value_keyPressed(e);
    }
   });
   I_R_FLabel.setFont(new Font("Dialog", 1, 15));
   I_R_FLabel.setText("Impulse Response File   :");
   F_R_FLabel.setFont(new Font("Dialog", 1, 15));
   F_R_FLabel.setText("Frequency response File :");
   F_R_File_Text.setBackground(Color.white);
   New_button.setLabel("New");
   New_button.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(MouseEvent e) {
      New_button_mouseClicked(e);
    }
   });
   Clear_button.setLabel("Clear");
   Cancell_button.setLabel("Cancel");
   Cancell_button.addMouseListener(new java.awt.event.MouseAdapter() {
public void mouseClicked(MouseEvent e) {
      Cancell_button_mouseClicked(e);
    }
   });
   Filter_type.setFont(new Font("Dialog", 1, 17));
   Filter_type.setText("Filter Type    : ");
   High_Pass_check.setFont(new Font("Dialog", 1, 15));
   High_Pass_check.setLabel("High Pass");
   High_Pass_check.setCheckboxGroup(Filter_Type_Group);
   High_Pass_check.addMouseListener(new java.awt.event.MouseAdapter() {
public void mouseClicked(MouseEvent e) {
      High_Pass_check_mouseClicked(e);
    }
   });
   Low_Pass_check.setCheckboxGroup(Filter_Type_Group);
   Status.setFont(new Font("Dialog", 1, 15));
   Status.setLabel("Status");
   Diplay_button.setLabel("Display");
   jProgressBar1.setForeground(Color.red);
   Diplay_button.addMouseListener(new java.awt.event.MouseAdapter() {
    public void mouseClicked(MouseEvent e) {
      Diplay_button_mouseClicked(e);
    }
   });

   Low_Pass_check.addMouseListener(new java.awt.event.MouseAdapter() {
```

```
public void mouseClicked(MouseEvent e) {
      Low_Pass_check_mouseClicked(e);
    }
  });
  Low_Pass_check.setLabel("Low Pass");
  this.setBackground(Color.darkGray);
  this.setFont(new Font("Dialog", 1, 15));
  xYLayout1.setWidth(875);
  this.getContentPane().setLayout(xYLayout1);
  this.setSize(800,600);
  this.getContentPane().add(Filter_type, new XYConstraints(25, 53, 117, 32));
  this.getContentPane().add(High_Pass_check, new XYConstraints(153, 101, 115, 33));
  this.getContentPane().add(Filter_ORder, new XYConstraints(27, 161, 116, 36));
  this.getContentPane().add(N1, new XYConstraints(150, 162, 30, 30));
  this.getContentPane().add(N2, new XYConstraints(200, 163, 30, 30));
  this.getContentPane().add(N3, new XYConstraints(246, 164, 30, 30));
  this.getContentPane().add(X1, new XYConstraints(183, 163, 14, 30));
  this.getContentPane().add(X2, new XYConstraints(232, 163, 14, 30));
  this.getContentPane().add(Cut_Off_Label, new XYConstraints(26, 236, 144, 35));
  this.getContentPane().add(Cut_Off_Value, new XYConstraints(172, 243, 31, 29));
  this.getContentPane().add(I_R_FLabel, new XYConstraints(34, 311, 184, 40));
  this.getContentPane().add(I_R_File_Text, new XYConstraints(223, 319, 158, 30));
  this.getContentPane().add(F_R_FLabel, new XYConstraints(30, 377, 189, 37));
  this.getContentPane().add(F_R_File_Text, new XYConstraints(223, 378, 156, 33));
  this.getContentPane().add(New_button, new XYConstraints(133, 469, 94, 40));
  this.getContentPane().add(Clear_button, new XYConstraints(269, 469, 98, 42));
  this.getContentPane().add(Cancell_button, new XYConstraints(566, 471, 91, 43));
  this.getContentPane().add(Low_Pass_check, new XYConstraints(154, 58, -1, 29));
  this.getContentPane().add(label2, new XYConstraints(307, 54, 290, 40));
  this.getContentPane().add(label1, new XYConstraints(307, 94, 290, 40));
  this.getContentPane().add(label3, new XYConstraints(307, 134, 290, 40));
  this.getContentPane().add(Status, new XYConstraints(298, 37, 308, 147));
  this.getContentPane().add(Diplay_button, new XYConstraints(408, 473, 111, 43));
  this.getContentPane().add(jProgressBar1, new XYConstraints(392, 321, 187, 23));
 }
public void start() {
 }
public void stop() {
 }
public void destroy() { }
public String getAppletInfo() {
 return "Applet Information";
 }
```

```
public String[][] getParameterInfo() {
  return null;
}
public static void main(String[] args) {
  Main applet = new Main();
  applet.isStandalone = true;
  JFrame frame = new JFrame();
  frame.setTitle("Design Of 3-D Filters Using Numirical Integration");
  frame.getContentPane().add(applet, BorderLayout.CENTER);
  applet.init();
  applet.start();
  frame.setSize(700,600);
  Dimension d = Toolkit.getDefaultToolkit().getScreenSize();
  frame.setLocation((d.width - frame.getSize().width) / 2, (d.height - frame.getSize().height) / 2);
  frame.setVisible(true);
}

void Low_Pass_check_mouseClicked(MouseEvent e) {
    label2.setForeground(Color.red);
   label2.setFont(new Font("Dialog", 0, 12));
  label2.setText("Designing a 3-D Low Pass FIR Filter");
}

void High_Pass_check_mouseClicked(MouseEvent e) {
 label2.setFont(new Font("Dialog", 0, 12));
 label2.setForeground(Color.red);
  label2.setText("Designing a 3-D High Passs FIR Filter");
  F_Type=2;
}

void N3_keyReleased(KeyEvent e) {
 label1.setFont(new Font("Dialog", 0, 12));
  label1.setForeground(Color.red);
  label1.setText("With Order of "+N1.getText()+" X "+N2.getText()+" X "+N3.getText());
}

void Cut_Off_Value_keyPressed(KeyEvent e) {
 label3.setFont(new Font("Dialog", 0, 12));
 label3.setForeground(Color.red);
  label3.setText("Cut off frequency of  "+Cut_Off_Value.getText());
}
void Cancell_button_mouseClicked(MouseEvent e) {
   System.exit(0);
```

```
  }
//------------
void New_button_mouseClicked(MouseEvent e) {
  Open_Files();
  num1=Integer.parseInt(N1.getText());
  num2=Integer.parseInt(N2.getText());
  num3=Integer.parseInt(N3.getText());
  jProgressBar1.setMaximum(num3*num2*num1);
  Impulse=new double[num1][num2][num3];

  F=new Filter(num1,num2,num3,F_Type);

  F.Integ((Double.valueOf(Cut_Off_Value.getText()).doubleValue()) );
  //--------------------------------
  for(I=0;I<num1;I++)
  for(J=0;J<num2;J++)
  for (K=0;K<num3;K++)
  {
  Impulse[I][J][K]=F.Impulse_Value(I,J,K);
  }

  Write_Impulse();
  }
//-------------------------------------------------------------------------
 void Diplay_button_mouseClicked(MouseEvent e)
  {

   JFrame Ch_Frame =new File_Chooser();
   Ch_Frame.addWindowListener(new WindowAdapter(){
     public void windowClosing(WindowEvent e){//System.exit(1);//
     }
     });
   Ch_Frame.pack();
   Ch_Frame.setVisible(true);
   Ch_Frame.setSize(250,250);
  }


//-------

void Open_Files()
{
 try
```

```
  {
        I_File=new RandomAccessFile
("c:/jbuilder2/myprojects/Simson/Data/IP.dat","rw");
  F_File=new RandomAccessFile     ("c:/jbuilder2/myprojects/Simson/Data/IP.dat","rw");
  }
catch(IOException ev)
  {
        System.err.println("File Could Not Be Opened\n"+ev.toString());
        System.exit(1);
  }
}

//-------------------------------------
public void Write_Impulse()
{
try{
     I_File.writeInt(num1);
     I_File.writeInt(num2);
     I_File.writeInt(num3);
     for(I=0;I<num1;I++)
     for(J=0;J<num2;J++)
     for (K=0;K<num3;K++)
     {
       jProgressBar1.setValue(jProgressBar1.getValue()+1);
       I_File.writeDouble(Impulse[I][J][K]);
     }
          I_File.close();
    }
    catch(IOException e)
    {
        System.err.println("File Could Not Be Opened\n"+e.toString());
          System.exit(1);
  }

}
//-------------------------------------

}
```

```
//Title:      Simson Rule of Integration
//Version:
//Copyright:  Copyright (c) 1999
//Author:     Idris S. EL-Feghi
//Company:    University Of Windsor
//Description: Three Dimensional Integration
package Simson;

import java.awt.*;
import java.awt.event.*;
import java.applet.*;
import borland.jbcl.layout.*;
import borland.jbcl.control.*;

public class Applet1 extends Applet {
  XYLayout xYLayout1 = new XYLayout();
  boolean isStandalone = false;
  LabelControl Filter_Order = new LabelControl();
  FieldControl N1 = new FieldControl();
  FieldControl N2 = new FieldControl();
  FieldControl N3 = new FieldControl();
  CheckboxControl Low_Pass = new CheckboxControl();
  LabelControl Filter_Type = new LabelControl();
  LabelControl labelControl2 = new LabelControl();
  LabelControl labelControl4 = new LabelControl();
  CheckboxControl High_Pass = new CheckboxControl();
  LabelControl Cut_Off_Label = new LabelControl();
  FieldControl Cut_Off_Value = new FieldControl();
  LabelControl I_R_Label = new LabelControl();
  LabelControl F_R_Label = new LabelControl();
  FieldControl I_R_Text = new FieldControl();
  FieldControl F_R_Text = new FieldControl();
  ButtonControl New_Button = new ButtonControl();
  ButtonControl Cancell_Button = new ButtonControl();
  CheckboxGroup checkboxGroup1 = new CheckboxGroup();
  Label FR = new Label();

  //Construct the applet
  public Applet1() {
  }

  //Initialize the applet
```

```
public void init() {
 try { jbInit(); } catch (Exception e) { e.printStackTrace(); }
}

//Component initialization
public void jbInit() throws Exception{
 xYLayout1.setWidth(525);
 xYLayout1.setHeight(494);
 Filter_Order.setFont(new Font("Dialog", 1, 12));
 Filter_Order.setText("Filter Order :");
 N1.setPostOnFocusLost(false);
 N1.addActionListener(new java.awt.event.ActionListener() {
  public void actionPerformed(ActionEvent e) {
   N1_actionPerformed(e);
  }
 });
 N2.setPostOnFocusLost(false);
 N2.setPostOnEndEdit(false);
 N3.setPostOnFocusLost(false);
 N3.setPostOnEndEdit(false);
 Low_Pass.setCheckboxGroup(checkboxGroup1);
 Low_Pass.setFont(new Font("Dialog", 3, 12));
 Low_Pass.setLabel("Low Pass");
 Filter_Type.setFont(new Font("Dialog", 1, 12));
 Filter_Type.setText("Filter Type   :");
 labelControl2.setFont(new Font("Dialog", 1, 15));
 labelControl2.setText("X");
 labelControl4.setFont(new Font("Dialog", 1, 15));
 labelControl4.setText("X");
 High_Pass.setCheckboxGroup(checkboxGroup1);
 High_Pass.setFont(new Font("Dialog", 3, 12));
 High_Pass.setLabel("High Pass");
 Cut_Off_Label.setFont(new Font("Dialog", 1, 12));
 Cut_Off_Label.setText("Cut Off Frequency :");
 Cut_Off_Value.setPostOnFocusLost(false);
 I_R_Label.setFont(new Font("Dialog", 1, 12));
 I_R_Label.setText("Impulse Response File     :");
 F_R_Label.setFont(new Font("Dialog", 1, 12));
 F_R_Label.setText("Frequency Response File :");
 I_R_Text.setColumnName("");
 I_R_Text.setPostOnFocusLost(false);
 I_R_Text.setSelected(true);
 F_R_Text.setPostOnFocusLost(false);
```

```
 New_Button.setLabel("New");
 New_Button.addMouseListener(new java.awt.event.MouseAdapter() {
  public void mouseClicked(MouseEvent e) {
   New_Button_mouseClicked(e);
  }
 });
 New_Button.addActionListener(new Applet1_New_Button_actionAdapter(this));
 Cancell_Button.setLabel("Cancel");
 Cancell_Button.addMouseListener(new java.awt.event.MouseAdapter() {
  public void mouseClicked(MouseEvent e) {
   Cancell_Button_mouseClicked(e);
  }
 });
 Cancell_Button.addActionListener(new Applet1_Cancell_Button_actionAdapter(this));


 this.setLayout(xYLayout1);
 this.add(Filter_Order, new XYConstraints(28, 36, 78, 25));
 this.add(N1, new XYConstraints(147, 36, 32, -1));
 this.add(N2, new XYConstraints(195, 36, 30, 24));
 this.add(N3, new XYConstraints(242, 38, 28, -1));
 this.add(Low_Pass, new XYConstraints(125, 84, 118, 18));
 this.add(Filter_Type, new XYConstraints(26, 68, 79, 37));
 this.add(labelControl2, new XYConstraints(182, 37, 11, -1));
 this.add(labelControl4, new XYConstraints(228, 36, 15, 29));
 this.add(High_Pass, new XYConstraints(126, 113, 90, 22));
 this.add(Cut_Off_Label, new XYConstraints(25, 163, -1, 24));
 this.add(Cut_Off_Value, new XYConstraints(158, 162, 42, -1));
 this.add(I_R_Label, new XYConstraints(26, 209, -1, 33));
 this.add(F_R_Label, new XYConstraints(23, 244, 156, 27));
 this.add(I_R_Text, new XYConstraints(194, 214, 137, 24));
 this.add(F_R_Text, new XYConstraints(193, 248, 138, -1));
 this.add(New_Button, new XYConstraints(41, 315, 73, 32));
 this.add(Cancell_Button, new XYConstraints(169, 315, 56, 29));
 this.add(FR, new XYConstraints(25, 387, 448, 97));
}

//Start the applet
public void start() {
}

//Stop the applet
public void stop() {
```

```
}

//Destroy the applet
public void destroy() {
}

//Get Applet information
public String getAppletInfo() {
  return "Applet Information";
}

//Get parameter info
public String[][] getParameterInfo() {
  return null;
}

//Main method
static public void main(String[] args) {
  Applet1 applet = new Applet1();
  applet.isStandalone = true;
  DecoratedFrame frame = new DecoratedFrame();
  frame.setTitle("3-D Filter Design using Numirical integration");
  frame.add(applet, BorderLayout.CENTER);
  applet.init();
  applet.start();
  frame.pack();
  Dimension d = Toolkit.getDefaultToolkit().getScreenSize();
  frame.setLocation((d.width - frame.getSize().width) / 2, (d.height - frame.getSize().height) / 2);
  frame.setVisible(true);
}

void Cut_Off_Value_actionPerformed(ActionEvent e) {

}

void F_R_Text_actionPerformed(ActionEvent e) {

}

void New_Button_actionPerformed(ActionEvent e) {

}
```

```
void Cancell_Button_actionPerformed(ActionEvent e) {

}

void I_R_Text_actionPerformed(ActionEvent e) {

}

void fieldControl7_actionPerformed(ActionEvent e) {

}

void checkboxPanel2_actionPerformed(ActionEvent e) {

}

void N1_actionPerformed(ActionEvent e) {

}

void Cancell_Button_mouseClicked(MouseEvent e) {
  FR.setText(I_R_Text.getText());

}

void New_Button_mouseClicked(MouseEvent e) {
  FR.setText("Filter to be designed with the Following specifications :");
  FR.setText("sajjfhfjf");
 }
}




}

class Applet1_fieldControl6_actionAdapter implements java.awt.event.ActionListener {
 Applet1 adaptee;

 Applet1_fieldControl6_actionAdapter(Applet1 adaptee) {
  this.adaptee = adaptee;
 }
```

```
}

class Applet1_New_Button_actionAdapter implements java.awt.event.ActionListener{
  Applet1 adaptee;

  Applet1_New_Button_actionAdapter(Applet1 adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.New_Button_actionPerformed(e);
  }
}

class Applet1_Cancell_Button_actionAdapter implements java.awt.event.ActionListener{
  Applet1 adaptee;

  Applet1_Cancell_Button_actionAdapter(Applet1 adaptee) {
    this.adaptee = adaptee;
  }

  public void actionPerformed(ActionEvent e) {
    adaptee.Cancell_Button_actionPerformed(e);
  }
}

class Applet1_fieldControl5_actionAdapter implements java.awt.event.ActionListener {
  Applet1 adaptee;

  Applet1_fieldControl5_actionAdapter(Applet1 adaptee) {
    this.adaptee = adaptee;
  }


}
```

```
//Title:       Simson Method For Numerical  Integration
//Version:
//Copyright:   Copyright (c) 1999
//Author:      Idris S. El-feghi
//Company:     university Of Windsor
//Description: Designing 3-D Filters using
//Simson's method for numerical
// integration

package Simson;

import java.awt.*;
import java.io.*;
import com.sun.java.swing.*;
import borland.jbcl.layout.*;
import java.awt.event.*;

public class Dialog1 extends JDialog{
//JDialog {
  JPanel panel1 = new JPanel();
  RandomAccessFile Display_File;
  String SS=new String();
  XYLayout xYLayout1 = new XYLayout();
//-----------------------------------------------

public Dialog1(Frame frame, String title, boolean modal) {
    super(frame, title, modal);
    //super(title);
    try  {
      SS=String.valueOf(title);
      Display_File=new RandomAccessFile     (SS,"r");
      frame.pack();
      frame.show();
      frame.setSize(100,100);

      jbInit();

    }
    catch (Exception ex) {
      ex.printStackTrace();
    }
  }
//-----------------------------------------------
```

```
public Dialog1()
  {
   this(null, "", false);

  }
//-----------------------------------------------
void jbInit() throws Exception {
   panel1.setLayout(xYLayout1);
   panel1.setRequestFocusEnabled(false);
   panel1.setMaximumSize(new Dimension(900, 900));
   panel1.setEnabled(true);
   panel1.setPreferredSize(new Dimension(600, 700));
   panel1.setBorder(BorderFactory.createEmptyBorder(30,30,10,30));
   getContentPane().add(panel1);
  }
//*****************************************************************************
 public void paint(Graphics g)
{
        int xpos=50,ypos=100,I,J,K,NS;
  double x1,x2,x3;
try
 {
  NS=Display_File.readInt();
  NS=Display_File.readInt();
  NS=Display_File.readInt();
  for(I=0;I<NS;I++)
   for(J=0;J<NS;J++)
    for(K=0;K<NS;K++)
          {
        g.drawString("I ("+I+","+J+","+K+") = "+Display_File.readDouble(),xpos,ypos);
        ypos+=20;
                }
         }
 catch(IOException ev)
   {
        System.err.println("Proble in reading The File\n"+ev.toString());
        System.exit(0);
   }
 }
 //*****************************************************************************
 }
```

```
//Title:       Simson Method For Numerical Integration
//Version:
//Copyright:   Copyright (c) 1999
//Author:      Idris S. El-feghi
//Company:     University Of Windsor
//Description: Designing 3-D Filters using
//Simson's method for numerical Function for calculating the impulse response
// integration


package Simson;
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
import java.io.*;
import com.sun.java.swing.*;
import borland.jbcl.layout.*;
import borland.jbcl.control.*;

public class Dsp_Class implements Runnable
{
public String FF;

public Dsp_Class(String File_Name)
 {
 FF=File_Name;
 }
public void run()
{}

//----------------------------
public void paint(Graphics g)
{
 RandomAccessFile I_File;
 try
  {
        I_File=new RandomAccessFile       ("c:/jbuilder2/myprojects/Simson/Data/"+FF,"r");
  }
 catch(IOException ev)
  {
        System.err.println("File Could Not Be Opened\n"+ev.toString());
        System.exit(1);
  }
```

```
 JFrame Cframe=new JFrame();
 Cframe.setTitle("Coefficients");
 Cframe.setSize(700,600);
 Dimension d = Toolkit.getDefaultToolkit().getScreenSize();
 Cframe.setLocation((d.width - Cframe.getSize().width) / 2, (d.height - Cframe.getSize().height)
/ 2);
 Cframe.setVisible(true);
 g.drawString("Done",100,20);
}
//--------------------------------------------------------------------------

//
}
```

```
//Title:       Simson Method For Numerical Integration
//Version:
//Copyright:   Copyright (c) 1999
//Author:      Idris S. El-feghi
//Company:     university Of Windsor
//Description: Designing 3-D Filters using
//Simson's method for numerical File Chooser  function
// integration

package Simson;

import java.io.*;
import java.awt.*;
import java.awt.event.*;
import com.sun.java.swing.*;
import com.sun.java.swing.preview.JFileChooser;


//-------------------------------------------------------------------------
public class File_Chooser extends JFrame
{
private JTextArea log;

private JFrame frame=new JFrame();

final JFileChooser FC=new JFileChooser();
public RandomAccessFile Display_File;
private JFrame Display_frame=new JFrame();
public Dialog1 DI;
private String newLine=System.getProperty("Line.separator");

public File_Chooser()
{
super("Choosing File For Dsiplay");

JButton OpenButton =new JButton("Open",new
ImageIcon("C:/jbuilder2/myprojects/Simson/Images/open.gif"));
JButton ExitButton =new JButton("Exit File",new
ImageIcon("C:/jbuilder2/myprojects/Simson/Images/open.gif"));

OpenButton.addActionListener(new OpenListener());
ExitButton.addActionListener(new ExitListener());
```

```
JPanel buttonpanel=new JPanel();
buttonpanel.add(OpenButton);
buttonpanel.add(ExitButton);

log=new JTextArea(5,20);
log.setMargin(new Insets(5,5,5,5));
JScrollPane logScrollPane=new JScrollPane(log);
Container contentPane=getContentPane();
contentPane.add(buttonpanel,BorderLayout.NORTH);
contentPane.add(logScrollPane,BorderLayout.CENTER);


}


//*-------------------------------------------------
private class OpenListener implements ActionListener
{
public void actionPerformed(ActionEvent e)
{
int RetVal=FC.showDialog(File_Chooser.this);
int xpos=50,ypos=100,I,J,K,NS;
double x1,x2,x3;

if(RetVal !=-1)
 {
  File file=FC.getSelectedFile();
  log.append("opening the file "+file.getName());
  //  "File Choose Error", JOptionPane.ERROR_MESSAGE);
   JFrame Display_Frame=new Dialog1(frame,file.getName(),true);
   Display_Frame.addWindowListener(new WindowAdapter(){
      public void windowClosing(WindowEvent e)
      {
      }
      });
    Display_Frame.pack();
    Display_Frame.setVisible(true);
    Display_Frame.setSize(250,450);

 }else
  {
  log.append("open Command cacIled by user.");
  }
```

```
  }
  }


//-----------------------------------------------------
private class ExitListener implements ActionListener
{
public void actionPerformed(ActionEvent e)
{



}
}
//-----------------------------------------------------
}
```

```
// Numerical integration function
package Simson;
public class Filter
{
private double   xmin,xmax,
                 ymin,ymax,
                              zmin,zmax,
          h[][][],d0,temp[],temp3[][];
private int     n1,n2,n3,
                          M,N,L,NS,Type;

final double    PI=3.1415926f;

//*----Class Constructer---*

public Filter(int na1,int na2,int na3,int T)
{
 xmin=ymin=zmin=0.0f;
 xmax=ymax=zmax=PI;
 M=N=L=80;
 Type=T;
  NS=(na1-1)/2;
  h=new double   [na1][na2][na3];
  temp=new double[NS+1];
  temp3=new double[NS*3][NS*3];


}
//*-----------------------------------------
public double fir(double x,double y,double z)
{
return H(x,y,z)*(double)(Math.cos((double)x*n1)*
                                          Math.cos((double)y*n2)*
                                          Math.cos((double)z*n3));
}
//*-----------------------------------------
public double H(double  x,double y,double z)
{
double R;
R=((x*x)+(y*y)+(z*z));
if(Type==1)
  return (0.414f*(d0*d0))/(R+(0.414f*d0*d0));
else
```

```
 return( R/(R+(0.414*(d0*d0) )) );



}
//*----------------------------------------
public double Simpson3()
{
double   dx,dy,dz,x,y,z,sum1,
       sum2,AR[][],VL[];
AR=new double [M+1][N+1];
VL=new double [M+1];
dx=(xmax-xmin)/M;
dy=(ymax-ymin)/N;
dz=(zmax-zmin)/L;
z=zmin;
for(int K=0;K<L;K++)
{
sum1=sum2=0.0f;
x=xmin;
for(int I=0;I<M;I++)
{
sum1=sum2=0.0f;
y=ymin+dy;
for(int J=1;J<N;J++)
  {if((J%2)==0)
    sum1+=fir(x,y,z);
   else
    sum2+=fir(x,y,z);
    y+=dy;
  }
  AR[K][I]=fir(x,y,zmin)+2.0*sum1+4.0*sum2+fir(x,y,zmax);
  x+=dx;
 }
 z+=dz;
 }
 //******_____**********
 for (int K=0;K<L;K++)
 {
 sum1=sum2=0.0;
 for(int I=0;I<M;I++)
 {
 if((I%2)==0)
```

```
 sum1+=AR[K][I];
else
 sum2+=AR[K][I];
}
VL[K]=(AR[K][0]+2.0*sum1+4.0*sum2+AR[K][M])*(dx*dy)/9.0f;
}

sum1=sum2=0;
for(int K=1;K<M;K++)
{
if((K%2)==0)
 sum1+=VL[K];
else
 sum2+=VL[K];
}
return (VL[0]+2.0*sum1+4.0*sum2+VL[M])*(dz)/3.0f;
}

//*-------------------------

public void Integ(double cutoff)
{
d0=cutoff*PI;

for (n1=0;n1<=NS;n1++)
for (n2=0;n2<=NS;n2++)
{if(n2 >n1) continue;
        for(n3=0;n3<=NS;n3++)
        h[n1][n2][n3]=Simpson3()/(PI*PI*PI);
}

for (n1=0;n1<=NS;n1++)
  for (n2=0;n2<=NS;n2++)
{       if(n2 < n1)
        for(n3=0;n3<=NS;n3++)
        h[n2][n1][n3]=h[n1][n2][n3];
}


//*----------Second Cube
for(n3=0;n3<=NS;n3++)
 temp[n3]=h[NS][NS][n3];
```

```
for (n1=NS;n1<((NS*2)+1);n1++)
 for (n2=NS;n2<((NS*2)+1);n2++)
    for (n3=0;n3<=NS;n3++)
                h[n1][n2][n3]=h[n1-NS][n2-NS][n3];
for(n3=0;n3<=NS;n3++)
 h[NS*2][NS*2][n3]=temp[n3];

//*----------Third Cube
 for (n1=NS;n1<((NS*2)+1);n1++)
  for (n2=0;n2<NS;n2++)
    for (n3=0;n3<=NS;n3++)
                h[n1][n2][n3]=h[n1][(NS*2)-n2][n3];
 //*----------Fourth Cube
 for (n1=0;n1< NS;n1++)
        for (n2=0; n2 < ((NS*2)+1);n2++)
     for (n3=0;n3<=NS;n3++)
                h[n1][n2][n3]=h[(NS*2)-n1][n2][n3];

//* ******************Second half***************
 for (n1=0;n1 < ((NS*2)+1);n1++)
        for (n2=0; n2 < ((NS*2)+1);n2++)
     for (n3=0;n3 < NS;n3++)
                h[n1][n2][(NS*2)-n3]=h[n1][n2][n3];
 //********************      */


}
//*-------------------------

public double Impulse_Value(int I,int J,int K)
{
  return h[I][J][K];
}
//*-------------------------

}
```

```
// Filter Class
package Simson;
public class Filter
{
private double   xmin,xmax,
                 ymin,ymax,
                              zmin,zmax,
          h[][][],d0,temp[],temp3[][];
private int     n1,n2,n3,
                          M,N,L,NS,Type;

final double    PI=3.1415926f;

//*----Class Constructer---*

public Filter(int na1,int na2,int na3,int T)
{
 xmin=ymin=zmin=0.0f;
 xmax=ymax=zmax=PI;
 M=N=L=80;
 Type=T;
   NS=(na1-1)/2;
   h=new double   [na1][na2][na3];
   temp=new double[NS+1];
   temp3=new double[NS*3][NS*3];


}
//*-----------------------------------------
public double fir(double x,double y,double z)
{
return H(x,y,z)*(double)(Math.cos((double)x*n1)*
                                        Math.cos((double)y*n2)*
                                        Math.cos((double)z*n3));
}
//*-----------------------------------------
public double H(double  x,double y,double z)
{
double R;
R=((x*x)+(y*y)+(z*z));
if(Type==1)
  return (0.414f*(d0*d0))/(R+(0.414f*d0*d0));
else
```

```
 return( R/(R+(0.414*(d0*d0) )) );



}
//*-----------------------------------------
public double Simpson3()
{
double   dx,dy,dz,x,y,z,sum1,
       sum2,AR[][],VL[];
AR=new double [M+1][N+1];
VL=new double [M+1];
dx=(xmax-xmin)/M;
dy=(ymax-ymin)/N;
dz=(zmax-zmin)/L;
z=zmin;
for(int K=0;K<L;K++)
{
sum1=sum2=0.0f;
x=xmin;
for(int I=0;I<M;I++)
{
sum1=sum2=0.0f;
y=ymin+dy;
for(int J=1;J<N;J++)
  {if((J%2)==0)
    sum1+=fir(x,y,z);
   else
    sum2+=fir(x,y,z);
    y+=dy;
  }
  AR[K][I]=fir(x,y,zmin)+2.0*sum1+4.0*sum2+fir(x,y,zmax);
  x+=dx;
 }
 z+=dz;
 }
 //******_____**********
 for (int K=0;K<L;K++)
 {
 sum1=sum2=0.0;
 for(int I=0;I<M;I++)
 {
 if((I%2)==0)
```

```
 sum1+=AR[K][I];
else
 sum2+=AR[K][I];
}
VL[K]=(AR[K][0]+2.0*sum1+4.0*sum2+AR[K][M])*(dx*dy)/9.0f;
}

sum1=sum2=0;
for(int K=1;K<M;K++)
{
if((K%2)==0)
 sum1+=VL[K];
else
 sum2+=VL[K];
}
return (VL[0]+2.0*sum1+4.0*sum2+VL[M])*(dz)/3.0f;
}

//*-------------------------

public void Integ(double cutoff)
{
d0=cutoff*PI;

for (n1=0;n1<=NS;n1++)
for (n2=0;n2<=NS;n2++)
{if(n2 >n1) continue;
        for(n3=0;n3<=NS;n3++)
        h[n1][n2][n3]=Simpson3()/(PI*PI*PI);
}

for (n1=0;n1<=NS;n1++)
  for (n2=0;n2<=NS;n2++)
{       if(n2 < n1)
        for(n3=0;n3<=NS;n3++)
        h[n2][n1][n3]=h[n1][n2][n3];
}


//*----------Second Cube
for(n3=0;n3<=NS;n3++)
 temp[n3]=h[NS][NS][n3];
```

```
for (n1=NS;n1<((NS*2)+1);n1++)
  for (n2=NS;n2<((NS*2)+1);n2++)
      for (n3=0;n3<=NS;n3++)
                h[n1][n2][n3]=h[n1-NS][n2-NS][n3];
for(n3=0;n3<=NS;n3++)
  h[NS*2][NS*2][n3]=temp[n3];

//*----------Third Cube
 for (n1=NS;n1<((NS*2)+1);n1++)
   for (n2=0;n2<NS;n2++)
     for (n3=0;n3<=NS;n3++)
                h[n1][n2][n3]=h[n1][(NS*2)-n2][n3];
 //*----------Fourth Cube
 for (n1=0;n1< NS;n1++)
         for (n2=0; n2 < ((NS*2)+1);n2++)
      for (n3=0;n3<=NS;n3++)
                h[n1][n2][n3]=h[(NS*2)-n1][n2][n3];

//* ******************Second half***************
 for (n1=0;n1 < ((NS*2)+1);n1++)
         for (n2=0; n2 < ((NS*2)+1);n2++)
      for (n3=0;n3 < NS;n3++)
                h[n1][n2][(NS*2)-n3]=h[n1][n2][n3];
 //*******************     */


}
//*-------------------------

public double Impulse_Value(int I,int J,int K)
{
  return h[I][J][K];
}
//*-------------------------

}
```

```
package Simson;


import java.awt.*;
import java.awt.event.*;
import java.io.*;
import com.sun.java.swing.*;
import borland.jbcl.layout.*;
import borland.jbcl.control.*;


public class Frame1 extends JFrame {
  RandomAccessFile Display_File;
  String SS=new String();
  XYLayout xYLayout1 = new XYLayout();

public Frame1()
{
}
//---------------------------------------
public Frame1(String S) {
    try {
      SS=String.valueOf(S);
          Display_File=new RandomAccessFile          (SS,"r");
      jbInit();
      }
    catch (Exception e) {
       e.printStackTrace();
    }
  }



 private void jbInit() throws Exception {
    this.setEnabled(true);
    this.setTitle(SS);
    this.getContentPane().setLayout(xYLayout1);
    this.setSize(800,600);
    this.setVisible(true);
    repaint();


  }
```

```
//*****************************************************************************
 public void paint(Graphics g)
{
        int xpos=50,ypos=50,I,J,K,NS;
 double x1,x2,x3;
try
 {

 NS=Display_File.readInt();
 NS=Display_File.readInt();
 NS=Display_File.readInt();
 for(I=0;I<NS;I++)
  for(J=0;J<NS;J++)
   for(K=0;K<NS;K++)
          {
        g.drawString("I ("+I+","+J+","+K+") = "+Display_File.readDouble(),xpos,ypos);
   ypos+=20;
                }

        }

catch(IOException ev)
  {
        System.err.println("Proble in reading The File\n"+ev.toString());
        System.exit(0);
  }


}
//-----------*/
 public synchronized void update(Graphics g)
{
repaint();}

//-----------*/

}
```

```
//This class for will be used for axis plotting
import java.awt.*;


public class Axis
{
public  int x0,y0,x1,y1,x2,y2,x3,y3,
        x4,y4,y5,x6,y6,WD;
public double unite;

public double xmin,xmax,ymin,ymax,zmin,zmax,
          C_max,C_min,Len;

public int R,G,B;
//constructer
public Axis(int height,int width,double AX,double BX,double AY,double BY,
                        double AZ,double BZ,double AC,double BC)
{
unite=height/20;
xmin=AX;
xmax=BX;
ymin=AY;
ymax=BY;
zmin=AZ;
zmax=BZ;
C_max=AC;
C_min=BC;
if(width>1000) WD=900;
else WD=600;

Len =Math.abs(C_max-C_min);

}

public double Unite()
{return unite;}



public void drawAxis(Graphics g,boolean B_Grid)
        {
        g.setColor(Color.black);
```

```
        x0=(int)(3*unite);//2
        y0=(int)(7*unite);//3

        x1=x0;
        y1=(int)(14*unite);//10

        x2=(int)(10.45*unite);//9
        y2=(int)(17.15*unite);//13

        x3=(int)(17.95*unite);//17
        y3=(int)(14.7*unite);//10

        x4=(int)(10.6*unite);//10
        y4=(int)(11.7*unite);//7

        y5=(int)(4.7*unite);//0

        x6=x3;
        y6=(int)(y0+(0.5*unite));//0


        g.drawLine(x0,y0,x1,y1);
        g.drawLine(x1,y1,x2,y2);
        g.drawLine(x2,y2,x3,y3);
        g.drawLine(x3,y3,x4,y4);
        g.drawLine(x4,y4,x1,y1);
        g.drawLine(x4,y4,x4,y5);
        g.drawLine(x4,y5,x0,y0);
        g.drawLine(x3,y3,x6,y6);
        g.drawLine(x6,y6,x4,y5);

    if(B_Grid)
        Grid(unite,g);
//      colorbar(g);
        }
        //*-------------------------
        public void Grid(double unite,Graphics g)
    {
        double xa,ya,za;
        g.setColor(Color.red);
        xa=0.0;ya=0.0;za=0.0;
        //----------------------------------------
         xa=0;
```

```
for(int J=1;J<7;J++)
{
 za=0;
 for(int K=0;K<40;K++)
 {
  g.drawString(".",DX(xa,J,za),DY(xa,J,za));
  za=za+0.2;
 }
}

//----------------------------------------
xa=0;
for(int J=1;J<8;J++)   {
 ya=0;
 for(int K=0;K<35;K++)
 {
  g.drawString(".",DX(xa,ya,J),DY(xa,ya,J));
  ya+=0.2;
 }
}

///---------------------------------------
ya=0;
for(int J=1;J<6;J++)//8
{
 za=0;
 for(int K=0;K<40;K++)
 {
  g.drawString(".",DX(J,ya,za),DY(J,ya,za));
  za=za+0.2;
 }
}
//---------------------------------------
xa=0;
for(int J=1;J<8;J++)//9
{
 xa=0;
 for(int K=1;K<40;K++)
 {
  g.drawString(".",DX(xa,ya,J),DY(xa,ya,J));
  xa+=0.2;
 }
}
```

```
        //----------------------------------------
        za=8;ya=0;
        for(int J=1;J<7;J++)
        {
         xa=0;
         for(int K=0;K<40;K++)
         {
          g.drawString(".",DX(xa,J,za),DY(xa,J,za));
          xa+=0.2;
         }
        }
        //-------------
        za=8;xa=0;
        for(int J=1;J<8;J++)
        {
         ya=0;
         for(int K=1;K<=35;K++)
         {
          g.drawString(".",DX(J,ya,za),DY(J,ya,za));
          ya+=0.2;
         }
        }
        //----------------------------------------
       }
       //*-------------------
public int DX(double xa,double ya,double za)
       {

      ya=ya*Unite();
      xa=xa*Unite();
      za=za*Unite();


      return (int)(3*Unite()+(xa*Math.cos(rad(24.2)))+(za*(Math.cos(rad(17.5)))));

      }
       //------
      public int DY(double xa,double ya,double za)
      {
      ya=ya*Unite();
      xa=xa*Unite();
      za=za*Unite();
```

```
return (int)(14*Unite()+(xa*Math.sin(rad(24.2)))-ya-(za*(Math.sin(rad(17.5)))));
}

//*---------------------
public double rad(double teta)
{return (teta*(22/7)/180);}

//*---------------------
public void Put_Labels(Graphics g)
{
 double ylabel,xlabel,zlabel;
 String S1=new String("");

 ylabel=ymin;
 xlabel=xmin;
 zlabel=zmin;
 S1=String.valueOf(ymin);
 for(int J=0;J<=7;J++)
 {
  if(S1.length() > 4) S1=(String.valueOf(ylabel)).substring( 0,4);
  if(ylabel>=0.15)
  g.drawString(S1,DX(0,J,-1),DY(0,J,-1)); //00
  ylabel+=Math.abs(ymax-ymin)/7;
  S1=String.valueOf(ylabel);
 }

 for(int J=0;J<=8;J++)
 {
if(Math.abs(xlabel) <=0.001)
S1=String.valueOf(0);
else S1=String.valueOf(xlabel);

  if(S1.length() > 4) S1=(String.valueOf(xlabel)).substring( 0,4);
   g.drawString(S1,DX(J,0,-1),DY(J,0,-1));
  xlabel+=Math.abs(xmax-xmin)/8;
 }

 for(int J=0;J<=8;J++)
 {
  S1=String.valueOf(zlabel);
  if(S1.length() > 4) S1=(String.valueOf(zlabel)).substring( 0,4);
```

```
            g.drawString(S1,DX(8.5,0,J-0.5),DY(8.5,0,J-0.5));
            zlabel+=Math.abs(zmax-zmin)/8;
           }



         }

//*********----------------------------------------
 public void colorbar(Graphics g)
  {
  int I,J,T,R;
  double y;
  String BLabel;
  g.setColor(Color.black);
  g.drawRect(WD,50,31,241);
  for (I=0;I<=4;I++)
  {
  BLabel=String.valueOf(C_max-(I*(Len/4)));
  g.drawString(BLabel,WD-25-(BLabel.length()),50+(I*60));
  g.drawString("_",WD-5,50+(I*60));
  }
  y=C_min;
  I=50;
  do{


if( y >= (C_max-Len/4) && y <= C_max )
          {
          T=(int)((520/Len)*(y-C_max+(0.49*Len))+0.5);
          R=(int)(167.69-(T*0.4615));
          g.setColor(new Color(T,0,0));
    }
else if( y >= (C_max-Len/2) && y < (C_max-Len/4))
          {T=(int)((520/Len)*(y-(C_max-Len/4)+(0.4915*Len))+0.5);
          R=(int)(227.69-(T*0.4615));
          g.setColor(new Color(255,T,0));
          }
else if(  y >= (C_max-(3*Len/4)) && y <(C_max-Len/2))
          {T=(int)((-520/Len)*(y-0.49*Len-(C_max-0.75*Len))+0.5);
          R=(int)(287.69-(T*0.4615));
          g.setColor(new Color(0,T,0));
          }
```

```
else //if ( y >= C_min  && y < (C_max-(3*(Len/4))) )

        {T=(int)((-520/Len)*(C_max-1.24*Len-y)+0.5);
        R=(int)(347.69-(T*0.46));
        g.setColor(new Color(0,0,T));
        }

        g.drawString("_____",WD+1,R);
        y+=0.01;
}while (y < C_max);

 }

//**********___
 public void SET_COLOR(double yx)
 {

if( yx >= (C_max-Len/4) && yx <= C_max )
        {
        R=(int)((-520/Len)*(C_max-yx-(0.49*Len))+0.5);
        G=0;
        B=0;
   }
else if( yx >= (C_max-Len/2) && yx < (C_max-Len/4))
        {
         R=255;
    B=0;
         G=(int)((-520/Len)*((C_max-(Len/4))-yx-(0.49*Len))+0.5);
        }

 else if(  yx >= (C_max-(3*Len/4)) && yx <(C_max-Len/2))
        {R=0;
         G=(int)((-520/Len)*((C_max-(Len/2))-yx-(0.49*Len))+0.5);
         B=0;
   }

 else
        {
        R=0;
        G=0;
        B=(int)((-520/Len)*((C_max-0.75*Len)-yx-(0.49*Len))+0.5);
    }
}
```

```
// This Applet is for plotting the impulse response
import java.applet.*;
import java.awt.*;
import java.io.*;

import DrawFrame;

public class Dimpls extends Applet
{
        boolean m_fStandAlone = false;
        boolean B_Grid=true;
        private Dimension d;
        private int SResolution;
        private Axis A;
        private double PI=3.141592654;
        RandomAccessFile F_File,I_File;
        private int IMax,JMax,KMax,IM,JM,KM;
        private double W1[],W2[],W3[],h[][][];



        public static void main(String args[])
        {
                DrawFrame frame = new DrawFrame("3D Axis");
                frame.show();
      frame.hide();
                frame.resize(frame.insets().left + frame.insets().right  + 800,
                                        frame.insets().top  + frame.insets().bottom + 400);

                Dimpls applet_Dimpls = new Dimpls();

                frame.add("Center", applet_Dimpls);
                applet_Dimpls.m_fStandAlone = true;
                applet_Dimpls.init();
                applet_Dimpls.start();
      frame.show();
        }
        //***********
        public Dimpls()
        {       }
        //***********
        public void init()
```

```
        {
  resize(800, 400);
        d=Toolkit.getDefaultToolkit().getScreenSize();
        try
  {

        I_File  =new RandomAccessFile      (("c:/Idris/Data/IIP.dat"),"r");

        IMax=I_File.readInt();
        JMax=I_File.readInt();
        KMax=I_File.readInt();
        h=new double[IMax][JMax][KMax];
        W1=new double[IMax];
        W2=new double[JMax];
        W3=new double[KMax];

        W1[0]=0;
        W2[0]=0;
        W3[0]=0;

        for(int I=1;I< IMax;I++)
                W1[I]=W1[I-1]+1;
        for(int J=1;J< JMax;J++)
                W2[J]=W2[J-1]+1;
        for(int K=1;K< KMax;K++)
                W3[K]=W3[K-1];

        for(int K=0;K<KMax;K++)
        for(int I=0;I<IMax;I++)
                for(int J=0;J<JMax;J++)
                  h[I][J][K]=I_File.readDouble();

        //A=new Axis(d.height,d.width,W_min(1),W_max(1),W_min(4),W_max(4),
        //                        W_min(2),W_max(2),W_max(4),W_min(4));

A=new Axis(d.height,d.width,W_min(1),W_max(1),W_min(1),W_max(4),
                  W_min(2),33,30,0);
        }

catch(IOException e)
  {
        System.err.println("Proble in reading The File\n"+e.toString());
        System.exit(1);
```

```
  }
        }
//*****************************************************
        public double W_max(int M)
        {
        double Temp=-9000.00;
        double WOM[];
        if(M<=3)
        {
        if (M==1) WOM=W1;
        else if(M==2) WOM=W2;
        else  WOM=W3;
        for(int I=0;I<WOM.length;I++)
          {if(WOM[I] >Temp) Temp=WOM[I];}

        }else
                {
                for(int I=0;I< IMax;I++)
                 for(int J=0;J< JMax;J++)
                   for(int K=0;K< KMax;K++)
                    if(h[I][J][K] >Temp)
                      {
                            Temp=h[I][J][K];
                            IM=I;JM=J;KM=K;
                      }

                }

        return(Temp);
        }
//*****************************************************
        public double W_min(int M)
        {
        double Temp=9000.00;
        double WOM[];
        if(M<=3)
        {
        if (M==1) WOM=W1;
        else if(M==2) WOM=W2;
        else  WOM=W3;
        for(int I=0;I<WOM.length;I++)
        {if(WOM[I] <Temp)
           Temp=WOM[I];
```

```
        }
        }else
                {
                for(int I=0;I< IMax;I++)
                for(int J=0;J< JMax;J++)
                  for(int K=0;K< KMax;K++)
                   if(h[I][J][K] < Temp)
                         Temp=h[I][J][K];
                }
        return(Temp);
        }

//*************************************************************************************
        public void paint(Graphics g)
        {
        double x1=0,x2=0,y1=0,y2=0,z1=0,z2=0,xf,yf,zf;



        A.drawAxis(g,B_Grid);
        A.Put_Labels(g);

        int K=16;
   for(int I=1;I<IMax;I++)
                for(int J=0;J<JMax;J++)
   //     for(int K=0;K<KMax;K++)
             {A.SET_COLOR(h[I][J][K]);

plot3d(setx(W1[I-1]),sety(h[I-1][J][K]),setz(W2[J]),setx(W1[I]),sety(h[I][J][K]),setz(W2[J]),g);

             }


        for(int I=0;I<IMax;I++)
    for(int J=1;J<JMax;J++)
     //for(int K=0;K<KMax;K++)
             {A.SET_COLOR(h[I][J][K]);

plot3d(setx(W1[I]),sety(h[I][J-1][K]),setz(W2[J-1]),setx(W1[I]),sety(h[I][J][K]),setz(W2[J]),g);

             }

        }
```

```
//*************************************************************************************
        public void plot3d(double xa,double ya,double za,
                double xb,double yb,double zb,Graphics g)
        {
        g.setColor(new Color(A.R,A.G,A.B));
        g.drawLine(A.DX(xa,ya,za),A.DY(xa,ya,za),A.DX(xb,yb,zb),A.DY(xb,yb,zb));
        }
//*************************************************
 public double setx(double x1)
 {if(A.xmin < 0 ) return (A.xmax+x1)*(4/A.xmax);
  else return x1*(8/A.xmax);
 }
 //-----------------------------------------
 public double sety(double y1)
 {if(A.ymin < 0 ) return (7-(7*(A.ymax-y1))/(A.ymax-A.ymin) );
 else return y1*(7/A.ymax);
 }
//-----------------------------------------
 public double setz(double z1)
 {if(A.zmin < 0 ) return (A.zmax+z1)*(4.5/A.zmax);
  else return z1*(9/A.zmax);
 }
//-----------------------------------------

}
```

```
//This Applet for plotting the magnitude response
import java.applet.*;
import java.awt.*;
import java.io.*;

import DrawFrame;

public class Draw extends Applet
{
        boolean m_fStandAlone = false;
        boolean B_Grid=true;
        private Dimension d;
        private int SResolution;
        private Axis A;
        private double PI=Math.PI;
        RandomAccessFile F_File,P_File;
        private int IMax,JMax,KMax,IM,JM,KM;
        private double W1[],W2[],W3[],h[][][];



public static void main(String args[])
        {
                DrawFrame frame = new DrawFrame("3D Axis");
                frame.show();
      frame.hide();
                frame.resize(frame.insets().left + frame.insets().right  + 800,
                                        frame.insets().top  + frame.insets().bottom + 400);

                Draw applet_Draw = new Draw();

                frame.add("Center", applet_Draw);
                applet_Draw.m_fStandAlone = true;
                applet_Draw.init();
                applet_Draw.start();
      frame.show();
        }
//***********
public Draw()
        {
        /* this applet will do the 3-d plotting at different values of W3*/
        //it uses the class Axis and //
```

```
        }

//************/
public void init()
        {
   resize(800, 400);
        d=Toolkit.getDefaultToolkit().getScreenSize();
        try
  {
F_File  =new RandomAccessFile(("c:/Idris/Data/FRS.dat"),"r");


        IMax=F_File.readInt();
        JMax=F_File.readInt();
        KMax=F_File.readInt();



        h=new double[IMax][JMax][KMax];
        W1=new double[IMax];
        W2=new double[JMax];
        W3=new double[KMax];

        for(int I=0;I< IMax;I++)
                W1[I]=F_File.readDouble();
        for(int J=0;J< JMax;J++)
                W2[J]=F_File.readDouble();
        for(int K=0;K< KMax;K++)
                W3[K]=F_File.readDouble();



        for(int K=0;K<KMax;K++)
        for(int I=0;I<IMax;I++)
                for(int J=0;J<JMax;J++)
                  h[I][J][K]=Math.abs(F_File.readDouble());



   A=new Axis(d.height,d.width,W_min(1),W_max(1),W_min(4),W_max(4),
                        W_min(2),W_max(2),W_max(4),W_min(4));
```

```
   //A=new Axis(d.height,d.width,-PI,PI,0,1,-PI,PI,0,1);
        }

catch(IOException e)
  {
        System.err.println("Proble in reading The File\n"+e.toString());
        System.exit(1);
  }
        }
//*****************************************************
        public double W_max(int M)
        {
        double Temp=-9000.00;
        double WOM[];
        if(M<=3)
        {
        if (M==1) WOM=W1;
        else if(M==2) WOM=W2;
        else  WOM=W3;
        for(int I=0;I<WOM.length;I++)
          {if(WOM[I] >Temp) Temp=WOM[I];}

        }else
                {
                for(int K=0;K< KMax;K++)
                for(int I=0;I< IMax;I++)
                 for(int J=0;J< JMax;J++)

                   if(h[I][J][K] >Temp)
                     {
                          Temp=h[I][J][K];
                          IM=I;JM=J;KM=K;
                     }

                }

        return(Temp);
        }
//*****************************************************
        public double W_min(int M)
        {
        double Temp=9000.00;
        double WOM[];
```

```
        if(M<=3)
        {
        if (M==1) WOM=W1;
        else if(M==2) WOM=W2;
        else  WOM=W3;
        for(int I=0;I<WOM.length;I++)
        {if(WOM[I] <Temp)
           Temp=WOM[I];
        }
        }else
                {
                for(int K=0;K< KMax;K++)
                 for(int I=0;I< IMax;I++)
                 for(int J=0;J< JMax;J++)

                   if(h[I][J][K] < Temp)
                         Temp=h[I][J][K];
                }
        return(Temp);
        }

//*****************************************************************************************
        public void paint(Graphics g)
        {
        int I,J,K=0;
        double x=0;
        A.drawAxis(g,B_Grid);
        A.Put_Labels(g);

        for(K=0;K<KMax;K++)
        for(I=0;I<IMax;I++)
        for(J=1;J<JMax;J++)
          {if(((W3[K]))==(W3[23]))
           {
                A.SET_COLOR(h[I][J][K]);

plot3d(setx(W1[I]),sety(h[I][J-1][K]),setz(W2[J-1]),setx(W1[I]),sety(h[I][J][K]),setz(W2[J]),g);
                }
          }

        for(K=0;K<KMax;K++)
        for(I=1;I<IMax;I++)
        for(J=0;J<JMax;J++)
```

```
          {if(((W3[K]))==W3[23])
                {
           A.SET_COLOR(h[I][J][K]);

plot3d(setx(W1[I-1]),sety(h[I-1][J][K]),setz(W2[J]),setx(W1[I]),sety(h[I][J][K]),setz(W2[J]),g);
                }
          }

        g.drawString("W3="+W3[23],100,100);
        }
//*****************************************************************************
        public void plot3d(double xa,double ya,double za,
                double xb,double yb,double zb,Graphics g)
        {
        g.setColor(new Color(0,0,0));
        //g.setColor();
        g.drawLine(A.DX(xa,ya,za),A.DY(xa,ya,za),A.DX(xb,yb,zb),A.DY(xb,yb,zb));
        }
//*****************************************************************************
 public double setx(double x1)
 {if(A.xmin < 0 ) return (A.xmax+x1)*(4/A.xmax);
  else return   (8*(A.xmin-x1)/(A.xmin-A.xmax));

 }
//-----------------------------------------
 public double sety(double y1)
 {if(A.ymin < 0 ) return (7-(7*(A.ymax-y1))/(A.ymax-A.ymin) );
  else return  (7*(A.ymin-y1)/(A.ymin-A.ymax));
 }
//-----------------------------------------
 public double setz(double z1)
 {if(A.zmin < 0 ) return (A.zmax+z1)*(4/A.zmax);  //4.5
  else return   (8*(A.zmin-z1)/(A.zmin-A.zmax)); //9
 }
//-----------------------------------------
}
// Main Frame
import java.awt.*;

public class DrawFrame extends Frame
{
        public DrawFrame(String str)
        {
```

```
        super (str);
    }

    public boolean handleEvent(Event evt)
    {
        switch (evt.id)
        {
            case Event.WINDOW_DESTROY:
                dispose();
                System.exit(0);
                return true;

            default:
                return super.handleEvent(evt);
        }
    }
}
```

```
// Printing applet
import java.awt.geom.*;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.print.PrinterJob;
import java.awt.event.*;
import java.awt.*;
import javax.swing.*;
import java.awt.print.*;
import java.io.*;
public class Plot_Print extends JPanel implements Printable, ActionListener {


   final static JButton button = new JButton("Print");
   boolean B_Grid=true;
        private Dimension d;
        private int SResolution;
        private Axis A;
        private double PI=Math.PI;
        RandomAccessFile F_File;
        private int IMax,JMax,KMax,IM,JM,KM;
        private double W1[],W2[],W3[],h[][][];



//-----------------------------------------
public Plot_Print() {

        setBackground(Color.white);
   button.addActionListener(this);
        d=Toolkit.getDefaultToolkit().getScreenSize();
        try
  {
  F_File  =new RandomAccessFile(("c:/Java_s/F_FIR.dat"),"r");


        IMax=F_File.readInt();
        JMax=F_File.readInt();
        KMax=F_File.readInt();



        h=new double[IMax][JMax][KMax];
```

```
        W1=new double[IMax];
        W2=new double[JMax];
        W3=new double[KMax];

        for(int I=0;I< IMax;I++)
                W1[I]=F_File.readDouble();
        for(int J=0;J< JMax;J++)
                W2[J]=F_File.readDouble();
        for(int K=0;K< KMax;K++)
                W3[K]=F_File.readDouble();



        for(int K=0;K<KMax;K++)
        for(int I=0;I<IMax;I++)
                for(int J=0;J<JMax;J++)
                  h[I][J][K]=Math.abs(F_File.readDouble());



  // A=new Axis(d.height,d.width,W_min(1),W_max(1),W_min(4),W_max(4),
//                        W_min(2),W_max(2),W_max(4),W_min(4));

  A=new Axis(d.height,d.width,-PI,PI,0,1,-PI,PI,0,1);

        }

catch(IOException e)
  {
        System.err.println("Proble in reading The File\n"+e.toString());
        System.exit(1);
  }
        }


//-----------------------------------------

  public void actionPerformed(ActionEvent e) {
     if (e.getSource() instanceof JButton) {
     PrinterJob printJob = PrinterJob.getPrinterJob();
     printJob.setPrintable(this);
     if (printJob.printDialog()) {
```

```
        try {
           printJob.print();
        } catch (Exception ex) {
           ex.printStackTrace();
        }
      }
        }

}


//-----------------------------------------
public double W_max(int M)
        {
        double Temp=-9000.00;
        double WOM[];
        if(M<=3)
        {
        if (M==1) WOM=W1;
        else if(M==2) WOM=W2;
        else  WOM=W3;
        for(int I=0;I<WOM.length;I++)
          {if(WOM[I] >Temp) Temp=WOM[I];}

        }else
                {
                for(int K=0;K< KMax;K++)
                for(int I=0;I< IMax;I++)
                 for(int J=0;J< JMax;J++)

                   if(h[I][J][K] >Temp)
                     {
                          Temp=h[I][J][K];
                          IM=I;JM=J;KM=K;
                      }

                }

        return(Temp);
        }
//*************************************************
        public double W_min(int M)
```

```
        {
        double Temp=9000.00;
        double WOM[];
        if(M<=3)
        {
        if (M==1) WOM=W1;
        else if(M==2) WOM=W2;
        else  WOM=W3;
        for(int I=0;I<WOM.length;I++)
        {if(WOM[I] <Temp)
           Temp=WOM[I];
        }
        }else
                {
                for(int K=0;K< KMax;K++)
                 for(int I=0;I< IMax;I++)
                 for(int J=0;J< JMax;J++)

                   if(h[I][J][K] < Temp)
                         Temp=h[I][J][K];
                }
        return(Temp);
        }
//-----------------------------------------
   public void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2 = (Graphics2D) g;
        Display_Data(g2);
}
//*****************************************************************
public void Display_Data(Graphics2D g2)
{
        int I,J,K=0;
        double x=0;
        A.drawAxis(g2,B_Grid);
        A.Put_Labels(g2);
        K=0;
        for(K=0;K<KMax;K++)
        for(I=0;I<IMax;I++)
        for(J=1;J<JMax;J++)
          {if(((W3[K]))==(W3[16]))
           {
                A.SET_COLOR(h[I][J][K]);
```

```
                plot3d(setx(W1[I]),sety(h[I][J-1][K]),setz(W2[J-1]),setx(W1[I]),sety(h[I][J][K]),s
etz(W2[J]),g2);
                }
          }

        for(K=0;K<KMax;K++)
        for(I=1;I<IMax;I++)
        for(J=0;J<JMax;J++)
          {if((W3[K])==W3[16])
                {
           A.SET_COLOR(h[I][J][K]);
                plot3d(setx(W1[I-1]),sety(h[I-1][J][K]),setz(W2[J]),setx(W1[I]),sety(h[I][J][K]),s
etz(W2[J]),g2);
                }
          }

         g2.drawString("Figure 2.1 3-D plot of the magnitude response of low-pass filter at
W3="+W3[18],100,600);
        }

//*************************************************************************************
        public void plot3d(double xa,double ya,double za,
        double xb,double yb,double zb,Graphics g)
        {
        //g.setColor(new Color(A.R,A.G,A.B));
        g.setColor(Color.black);
        g.drawLine(A.DX(xa,ya,za),A.DY(xa,ya,za),A.DX(xb,yb,zb),A.DY(xb,yb,zb));
        }
//*************************************************************************************

  public double setx(double x1)
  {if(A.xmin < 0 ) return (A.xmax+x1)*(4/A.xmax);
   else return    (8*(A.xmin-x1)/(A.xmin-A.xmax));

  }
 //-----------------------------------------
  public double sety(double y1)
  {if(A.ymin < 0 ) return (7-(7*(A.ymax-y1))/(A.ymax-A.ymin) );
   else return  (7*(A.ymin-y1)/(A.ymin-A.ymax));

  }
//-----------------------------------------
  public double setz(double z1)
```

```
 {if(A.zmin < 0 ) return (A.zmax+z1)*(4.0/A.zmax);
  else return  (8*(A.zmin-z1)/(A.zmin-A.zmax));

 }
//-----------------------------------------



//*****************************************************************

   public int print(Graphics g, PageFormat pf, int pi) throws PrinterException {
      if (pi >= 1) {
         return Printable.NO_SUCH_PAGE;
         }
        Display_Data((Graphics2D) g);
      return Printable.PAGE_EXISTS;
   }

  public static void main(String s[]){
        WindowListener l = new WindowAdapter() {
                public void windowClosing(WindowEvent e) {System.exit(0);}
                public void windowClosed(WindowEvent e) {System.exit(0);}
        };
        JFrame f = new JFrame();
        f.addWindowListener(l);
        JPanel panel = new JPanel();
        panel.add(button);
        f.getContentPane().add(BorderLayout.SOUTH, panel);
        f.getContentPane().add(BorderLayout.CENTER, new Plot_Print());
        f.setSize(580, 500);
        f.show();
   }

}
```

APPENDIX B

DESIGN OF 3-D FIR FILTERS USING FFT AND WINDOW FUNCTIONS COMPUTER PROGRAMS

```
// Main Applet for the design of 3-D FIR filters
import java.applet.*;
import java.awt.*;
import java.io.*;
import firFrame;

public class fir extends Applet
{
        Thread          m_fir = null;
        private Graphics g;
        private WINGC3 WG;
        private int I=0,nums1,nums2,nums3,WType,FType;
        private Choice FT_Button,WT_Button;
        private Button B1,B2,B3;
        private TextField V_Cof,N1,N2,N3,V_alpha,TIF,TFF;
        private Label T_Type,T_Cof,T_Win,T_alpha,W_Size,X1,X2;
        private Label LIF,LFF;
        boolean m_fStandAlone = false;
        double  Val_Cof,Val_alpha;
        private MenuBar bar;
        private String SFT,SWT,FR,IR;
        private DrawFrame F;
//-------------------------------------------------------------------------
public static void main(String args[])
        {
                firFrame frame = new firFrame("Design Of 3D FIR Filters Using FFT and
Windows Functions");
                frame.show();
      frame.hide();
                frame.resize(frame.insets().left + frame.insets().right  + 600,
                                        frame.insets().top  + frame.insets().bottom + 400);

                fir applet_fir = new fir();

                frame.add("Center", applet_fir);
                applet_fir.m_fStandAlone = true;
                applet_fir.init();
      frame.show();
        }
//---------------------------------------------
        public fir()
        {
```

```
FT_Button= new Choice();
        WT_Button= new Choice();
        T_Type=new Label("Filter Type :");
        T_Cof =new Label("Cut Off Frequency :");
        V_Cof =new  TextField(5);
        T_Win =new Label("Window Function  ");
        T_alpha=new Label("Alpha :");
        V_alpha=new TextField(5);
        W_Size=new Label("Filter Order ");
        N1 =new  TextField(5);
        N2 =new  TextField(5);
        N3 =new  TextField(5);
        TIF=new  TextField(15);
        TFF=new  TextField(15);
        X1= new Label("X");
        X2= new Label("X");
        B1=new Button("New    ");
        B2=new Button("Cancel ");
        B3=new Button("Plot   ");



        SFT=new String();
        SWT=new String();


        LIF=new Label("File to Store Impulse  Response:");

        LFF=new Label("File to Store Magnitude Response:");



        FT_Button.addItem("ButterWorth HighPass      ");
        FT_Button.addItem("ButterWorth LowPass       ");
        FT_Button.addItem("Abrupt Transation HighPass");
        FT_Button.addItem("Abrupt Transation LowPass ");


        WT_Button.addItem("Rectangular Window ");
        WT_Button.addItem("Hanning    Window ");
        WT_Button.addItem("Hamming    Window ");
        WT_Button.addItem("Blackman   Window ");
        WT_Button.addItem("Kaiser     Window ");
```

```
            F=new DrawFrame("Plot");
        }

//-----------------------------------------------
        public void init()
        {
            setLayout(null);

            add(T_Type);
            add(FT_Button);
            add(T_Cof);
            add(V_Cof);
            add(T_Win);
            add(T_alpha);
            add(V_alpha);
            add(WT_Button);
            add(W_Size);
            add(N1);
            add(X1);
            add(N2);
            add(X2);
            add(N3);
            add(LIF);
            add(TIF);
            add(LFF);
            add(TFF);
            add(B1);
            add(B2);
            add(B3);



            T_Type.reshape(10,30,100,20);
            FT_Button.reshape(120,30,180,20);

            T_Cof.reshape(350,30,100,20);
            V_Cof.reshape(450,30,50,20);

            T_Win.reshape(10,80,100,20);
            WT_Button.reshape(120,80,180,20);

            T_alpha.reshape(380,80,50,20);
```

```
            V_alpha.reshape(440,80,50,20);

            W_Size.reshape(10,130,100,20);

            N1.reshape(130,130,30,20);
            X1.reshape(160,130,30,20);
            N2.reshape(190,130,30,20);
            X2.reshape(210,130,30,20);
            N3.reshape(240,130,30,20);

            LIF.reshape(10,180,180,20);
            TIF.reshape(200,180,100,20);

            LFF.reshape(10,230,180,20);
            TFF.reshape(200,230,100,20);

            B1.reshape(100,280,50,20);
            B2.reshape(250,280,50,20);
            B3.reshape(400,280,50,20);

        }
//********-----------------------------------
        public void destroy()
        {
        }
//*************************************************
        public void paint(Graphics g)
        {


        if(I==1)
          WG=new WINGC3(FType,Val_Cof,WType,Val_alpha,nums3,IR,FR,g);

         I=0;

        }
//*************************************************

public boolean action(Event e,Object o)
  {
  if(e.target== B2) System.exit(1);
  //---------
  if(e.target instanceof Choice)
```

```
{
FType=FT_Button.getSelectedIndex();
SFT=FT_Button.getSelectedItem();
}
//---------
if(e.target== V_Cof)
      {Double val=(Double.valueOf(e.arg.toString()));
 Val_Cof=val.doubleValue();
 }
//---------

if(e.target instanceof Choice)
 { WType=WT_Button.getSelectedIndex();
  SWT=WT_Button.getSelectedItem();

      }


if(e.target== V_alpha)
  { Double val=(Double.valueOf(e.arg.toString()));
   Val_alpha=val.doubleValue();
        }

 if(e.target==N1)
       nums1=Integer.parseInt(N1.getText());
      if(e.target==N2)
 nums2=Integer.parseInt(N2.getText());
      if(e.target==N3)
       nums3=Integer.parseInt(N3.getText());


      if(e.target==TIF)
       IR="c:/idris/Data/".concat(o.toString());

      if(e.target==TFF)
            FR="c:/idris/Data/".concat(o.toString());

      if(e.target==B2)
            System.exit(0);

      if(e.target==B1)
       {
            //showStatus("Still Calculating");
```

```
            I=1;

            //showStatus("Done Calculating");

        }
        /*

        if(e.target==B3)
        {
           MenuBar bar=new MenuBar();
                Menu Draw_Menu=new Menu("Plot");
                Draw_Menu.add("With W3= 0   ");
                Draw_Menu.add("contor Plot  ");
                Draw_Menu.add("Full Spher   ");
                Draw_Menu.add("1/4 Of Spher ");
                Draw_Menu.add("color Repre. ");
                bar.add(Draw_Menu);
           F.setMenuBar(bar);
           F.resize(300,100);
           F.show();
                //DR1=new Draw(g);
        }                                                    */

 //---------

  repaint();
  return true;
  }
//*************************************************

}

// FFT3DC Class used for computing 2-D FFT
public class FFT3DC
{
        //this class for 3-d FFT and IFFT
```

```
   public int L[],N;
        public double Factor,Wr[],Wi[],FR[][],FI[][],hr[][][],hi[][][];
        public int m;
        public double Rinter[][],Iinter[][],inr[][];
        public double HR[][][],HI[][][],HFR3[][][],HFI3[][][];
        public double xi[],xr[];
//---------Constructer-----------
public FFT3DC(double XR[][][],double XI[][][],int NL,int Sign)
{
N=NL;
L=new int[N];

hr=new double[N][N][N];
hi=new double[N][N][N];


HR=new double[N][N][N];
HI=new double[N][N][N];

HFR3=new double[N][N][N];
HFI3=new double[N][N][N];

FR=new double[N][N];
FI=new double[N][N];

Rinter=new double[N][N];
Iinter=new double[N][N];

xr=new double[N];
xi=new double[N];

Wr=new double[N];
Wi=new double[N];


m=(int)Math.round(Math.log((float)N)/Math.log(2.0f));
hr=XR;
hi=XI;
FFT3(Sign);
}

//-------------------------------
public double FFT3D_Out(int I,int J,int K,int RI)
```

```
{
if(RI==0)     return (HFR3[I][J][K]/Factor);
else  return (HFI3[I][J][K]/Factor);
}
//-------------------------
public  void FFT3(int Sign)
{
 int I,J,K;
 Bit_Reversal();
 TW_Factors(Sign);
 for(I=0;I<N;I++)
        {
        FFT2D(I);
        for(J=0;J<N;J++)
        for(K=0;K<N;K++)
        {
          HR[I][J][K]=Rinter[J][K];
          HI[I][J][K]=Iinter[J][K];
        }
        }

//************
        for(K=0;K<N;K++)
        {
        for(J=0;J<N;J++)
        {
        for(I=0;I<N;I++)
          {
           xr[L[I]]=HR[I][J][K];
           xi[L[I]]=HI[I][J][K];
          }
        FFT();

        for(I=0;I<N;I++)
        {
        HFR3[I][J][K]=xr[I];
        HFI3[I][J][K]=xi[I];
        }
        }
        }
}
//-------------------------
public void FFT2D(int LL)
```

```
{
        int I,J,K;
        for(I=0;I<N;I++)
        {
        for(K=0;K<N;K++)
          {
           xr[L[K]]=hr[LL][I][K];
           xi[L[K]]=hi[LL][I][K];
          }
        FFT();
        for(J=0;J<N;J++)
        {
        Rinter[I][J]=xr[J];
        Iinter[I][J]=xi[J];
        }
        }
        Transpose();
//*--------------------
        for(I=0;I<N;I++)
        {
        for(K=0;K<N;K++)
        {
         xr[L[K]]=Rinter[I][K];
         xi[L[K]]=Iinter[I][K];
        }
   FFT();
        for(J=0;J<N;J++)
        {
        Rinter[I][J]=xr[J];
        Iinter[I][J]=xi[J];
        }
        }
        Transpose();
}


//*******************************************
public void FFT()
{
int ip,k,kk,l,incr,iter,j,i;
double Tr,Ti;
ip=1;
kk=(N>>1);
```

```
incr=2;


for(iter=0;iter<m;iter++)
{
 for(j=0;j<N;j+=incr)
  {
  i=j+ip;
  Tr=xr[i];
  Ti=xi[i];

  xr[i]=xr[j]-Tr;
  xi[i]=xi[j]-Ti;

  xr[j]=xr[j]+Tr;
  xi[j]=xi[j]+Ti;
 }
 if(iter !=0)
   {
        for(k=1;k<ip;k++)
        {
        l=k*kk-1;
        for(j=k;j<N;j+=incr)
        {
                i=j+ip;
                Tr=xr[i]*Wr[l]-xi[i]*Wi[l];
                Ti=xr[i]*Wi[l]+xi[i]*Wr[l];

                xr[i]=xr[j]-Tr;
                xi[i]=xi[j]-Ti;

                xr[j]=xr[j]+Tr;
                xi[j]=xi[j]+Ti;
        }
        }
   }
  kk>>=1;
  ip<<=1;
  incr<<=1;
 }

}
//-------------------------
```

```
public void Transpose()
{

 int I,J;

 for(I=0;I<N;I++)
        for(J=0;J<N;J++)
              {
              FR[I][J]=Rinter[J][I];
              FI[I][J]=Iinter[J][I];
     }

 for(I=0;I<N;I++)
        for(J=0;J<N;J++)
              {
              Rinter[I][J]=FR[I][J];
              Iinter[I][J]=FI[I][J];
     }


}


//----------------------------
  private void Bit_Reversal()
        {

int Mask,C,A=0,J,K,I;



for(K=0;K<N;K++)
        {
        Mask=1;
        C=0;
        for(I=0,J=m-1;I<m;I++,J--)
         {
          A=(K&Mask)>>I;
               A<<=J;
          C|=A;
          Mask=Mask<<1;
         }
```

```
            L[K]=C;


          }
          }
//----------------end of bit reversal--------
  private void TW_Factors(int Sign)
  {
                if(Sign==0) Factor=1.0;
                else if(Sign==1) Factor=N*N*N;
   for(int I=0;I<((N>>1)-1);I++)
   {
        Wr[I]=((double)Math.cos((double)((I+1)*(2.0*Math.PI)/N)));
        Wi[I]=((double)Math.sin((double)((I+1)*(2.0*Math.PI)/N)));
        if(Sign==0) Wi[I]=(-1)*Wi[I];
        }
  }
//--------------------------
}//end of the FFT3DC class
```

```
//this class for generating the frequency response
// using fft3 and window functions
import java.awt.*;
import java.io.*;

public class WINGC3
{
        RandomAccessFile I_File,F_File;
        public double h[][][],CF,alpha;
        public double hr[][][],hi[][][];
        private String Ir,Fr;
        public FFT3DC F;
        public int Type,M1,N,WType,done=0;


 //--------------------
public WINGC3(int Filtype,double Cut_oF,int Window,double walpha,int Wd,String IR,String
FR,Graphics g)
        {
        Ir=IR;
        Fr=FR;
        Type=Filtype;
        CF=Cut_oF;
        N=Wd;
        WType=Window;
        alpha=walpha;
        M1=32;
        h=new double[17][17][17];
        hr=new double[65][65][65];
        hi=new double[65][65][65];
        try
          {
          I_File  =new RandomAccessFile  (Ir,"rw");
        F_File  =new RandomAccessFile    (Fr,"rw");
          }
        catch(IOException e)
    {
          System.err.println("Proble in Opening the Files\n"+e.toString());
          System.exit(1);
          }
        Compute(g);
}
        //--------------------
```

```
public int Fdone()
{
return(done);
}

//---------------------
        private void Compute(Graphics g)
        {
        int N1,M,M2,M3,I,J,K,i,j,k,l,m,n,II,JJ,KK,Nt;
        double theta,NSQRT,R2,NSQ,W[],xrm,xim,xrn,xin,zrt,mag,zit,dw,rl,im;
        double xrl=0,xil=0,ZX=0;

        M=M1>>1;
        CF=(CF/Math.PI)*M;
        CF*=CF;
        for(I=0;I<M1;I++)
        {
        II=(I-M)*(I-M);
        for(J=0;J<M1;J++)
        {
        JJ=(J-M)*(J-M);
        for(K=0;K<M1;K++)
        {
        KK=(K-M)*(K-M);
        R2=(double)(JJ+II+KK);
        hr[I][J][K]=Filter(R2,CF)*(Math.pow(-1,(I+J+K)));
        hi[I][J][K]=0.0;
        }
        }
        }

        F=new FFT3DC(hr,hi,M1,1);


//-----------------------------------------//

        N1=(N-1)/2;
        M2=M-N1;
        M3=M+N1;

        for(I=M2;I<=M3;I++)
                for(J=M2;J<=M3;J++)
                  for(K=M2;K<=M3;K++)
```

```
            h[I-M2][J-M2][K-M2]=(F.FFT3D_Out(I,J,K,0)*Math.pow(-1,I+J+K));
//-----------------------------------------//
        theta=(Math.PI)/((float)N1*Math.sqrt(3.0));
        for(I=0;I<N;I++)
        {
        II=(I-N1)*(I-N1);
        for(J=0;J<N;J++)
        {JJ=(J-N1)*(J-N1);
        for(K=0;K<N;K++)
        {
        KK=(K-N1)*(K-N1);
         NSQ=(double)(II+JJ+KK);
         NSQRT=Math.sqrt((double)NSQ) /(Math.sqrt((double)3.0));
         h[I][J][K]*=Window(theta,NSQRT,WType,alpha,N1);
        }
        }
        }


        //-------------
        try
        {
        I_File.writeInt(N);
        I_File.writeInt(N);
        I_File.writeInt(N);
        for(I=0;I<N;I++)
        for(J=0;J<N;J++)
        for(K=0;K<N;K++)
           I_File.writeDouble(h[I][J][K]);
           I_File.close();
        g.drawString(Ir+"  Impulse Response fie is Written",10,20);
        }
        catch(IOException e)
   {
        System.err.println("Proble in reading The File\n"+e.toString());
        System.exit(1);
   }

//*********************************
        try
        {
        M1=32;
        Nt=M1+1;
```

```
W=new double[Nt];
F_File.writeInt(Nt);
F_File.writeInt(Nt);
F_File.writeInt(Nt);
W=new double[Nt];
W[0]=-(Math.PI);
dw=2*Math.PI/M1;
for(I=1;I<Nt;I++)
        W[I]=W[I-1]+dw;

for(I=0;I<Nt;I++)
        F_File.writeDouble(W[I]);
for(I=0;I<Nt;I++)
        F_File.writeDouble(W[I]);
for(I=0;I<Nt;I++)
        F_File.writeDouble(W[I]);


for(m=0;m<Nt;m++)
{
for(n=0;n<Nt;n++)
for(l=0;l<Nt;l++)
{
zrt=0.0;zit=0.0;

for(i=-N1;i<=N1;i++)
for(j=-N1;j<=N1;j++)
for(k=-N1;k<=N1;k++)
{
        xrm=Math.cos((double)(i*W[m]));
        xim=Math.sin(-(double)(i*W[m]));

        xrn=Math.cos((double)(j*W[n]));
        xin=Math.sin(-(double)(j*W[n]));

        xrl=Math.cos((double)(k*W[l]));
        xil=Math.sin(-(double)(k*W[l]));

        rl=(xrm*xrn-xim*xin);
        im=(xrm*xin+xim*xrn);
```

```
                rl=rl*xrl-im*xil;
                im=rl*xil+im*xrl;

          zrt+=(h[k+N1][i+N1][j+N1])*rl;
                 zit+=(h[k+N1][i+N1][j+N1])*im;
          }
                mag=Math.sqrt((double)(zrt*zrt+zit*zit));
                F_File.writeDouble(mag);

                g.drawString("Writting the Frequency response ....",10,320);
          }
          }
          //---------
          done=1;
          F_File.close();
          g.drawString(Fr+"    Frequency Response fie is Written",300,20);
    }
          catch(IOException e)
      {
            System.err.println("Proble in reading The File\n"+e.toString());
            System.exit(1);
           }

  }
//------------------
          private double Filter(double R2,double CF)
          {
          if    (Type==0)  return(R2/(R2+0.414*CF));
          else if (Type==1)       return((0.414*CF)/(R2+0.414*CF));
          else {
                    if(Type==2)   //high pass abrupt transation
               {
                    if (R2 < CF)  return (0.0);
                    else return (1.0);
               }

           else            //low pass abrupt transation
               {
                    if (R2 < CF)  return (1.0);
                    else return (0.0);
               }
```

```
            }
        }

//-----------------------------------
private double Window(double theta,double NSQRT,int T,double alpha,int N1)
        {
        double beta,sum1,sum2,t;
        int K;
        if(T==0) return(1.0);
        else if(T==1)
                return(0.5+0.5*(Math.cos(theta*NSQRT)));
        else if(T==2)
                return(0.54+0.46*(Math.cos(theta*NSQRT)));
        else if(T==3)
return(0.42+0.5*(Math.cos(theta*NSQRT))+0.08*Math.cos(2.0*theta*NSQRT));
        else
          {
                alpha/=2.0;
                beta=(NSQRT/(float)N1)*(NSQRT/(float)N1);
                beta=alpha*Math.sqrt(Math.abs(1.0-beta));
                t=alpha;
                sum1=1.0+(t*t);
                for(K=2;K<11;K++)
                {
                 t=(1.0/(double)K)*alpha*t;
                 sum1+=t*t;
                                }
                t=beta;
                sum2=1.0+(t*t);
                for(K=2;K<11;K++)
                {
                 t=(1.0/(double)K)*beta*t;
                 sum2+=t*t;
                }
                return(sum2/sum1);
        }
        //---------
        }
}
```

```
// This class is for generating the impulse response of FIR filters
import java.awt.*;
import java.io.*;

public class WINGCI3
{
        RandomAccessFile I_File,F_File;
        public double h[][][],CF,alpha;
        public double hr[][][],hi[][][];
        private String Ir,Fr;
        public FFT3DC F;
        public int Type,M1,N,WType,done=0;


 //--------------------
public WINGCI3(int Filtype,double Cut_oF,int Window,double walpha,int Wd,String IR,String
FR,Graphics g)
        {
        Ir=IR;
        Fr=FR;
        Type=Filtype;
        CF=Cut_oF;
        N=Wd;
        WType=Window;
        alpha=walpha;
        M1=32;
        h=new double[17][17][17];
        hr=new double[65][65][65];
        hi=new double[65][65][65];
        try
          {
          I_File  =new RandomAccessFile  (Ir,"rw");
        F_File  =new RandomAccessFile    (Fr,"rw");
          }
        catch(IOException e)
    {
          System.err.println("Proble in Opening the Files\n"+e.toString());
          System.exit(1);
          }
        Compute(g);
}
        //--------------------
  public int Fdone()
```

```
{
return(done);
}

//---------------------
        private void Compute(Graphics g)
        {
        int N1,M,M2,M3,I,J,K,i,j,k,l,m,n,II,JJ,KK,Nt;
        double theta,NSQRT,R2,NSQ,W[],xrm,xim,xrn,xin,zrt,mag,zit,dw,rl,im;
        double xrl=0,xil=0,ZX=0;
        W=new double[M1+1];
        M=M1>>1;
        CF=(CF/Math.PI)*M;
        CF*=CF;
        for(I=0;I<M1;I++)
        {
        II=(I-M)*(I-M);
        for(J=0;J<M1;J++)
        {
        JJ=(J-M)*(J-M);
        for(K=0;K<M1;K++)
        {
        KK=(K-M)*(K-M);
        R2=(double)(JJ+II+KK);
        hr[I][J][K]=Filter(R2,CF)*(Math.pow(-1,(I+J+K)));
        hi[I][J][K]=0.0;
        }
        }
        }

        F=new FFT3DC(hr,hi,M1,1);


//-----------------------------------------//

        N1=(N-1)/2;
        M2=M-N1;
        M3=M+N1;

        for(I=M2;I<=M3;I++)
                for(J=M2;J<=M3;J++)
                  for(K=M2;K<=M3;K++)
            h[I-M2][J-M2][K-M2]=(F.FFT3D_Out(I,J,K,0)*Math.pow(-1,I+J+K));
```

```
//-----------------------------------------//
        theta=(Math.PI)/((float)N1*Math.sqrt(3.0));
        for(I=0;I<N;I++)
        {
        II=(I-N1)*(I-N1);
        for(J=0;J<N;J++)
        {JJ=(J-N1)*(J-N1);
        for(K=0;K<N;K++)
        {
        KK=(K-N1)*(K-N1);
         NSQ=(double)(II+JJ+KK);
         NSQRT=Math.sqrt((double)NSQ) /(Math.sqrt((double)3.0));
         h[I][J][K]*=Window(theta,NSQRT,WType,alpha,N1);
        }
        }
        }


        //------------
        try
        {
        I_File.writeInt(N);
        I_File.writeInt(N);
        I_File.writeInt(N);
        for(I=0;I<N;I++)
        for(J=0;J<N;J++)
        for(K=0;K<N;K++)
           I_File.writeDouble(h[I][J][K]);
           I_File.close();
        g.drawString(Ir+"  Impulse Response fie is Written",10,20);
        }
        catch(IOException e)
   {
        System.err.println("Proble in reading The File\n"+e.toString());
        System.exit(1);
   }

//*******************************
        try
        {
        Nt=M1+1;
        F_File.writeInt(Nt);
        F_File.writeInt(Nt);
```

```
F_File.writeInt(Nt);
W=new double[Nt];
W[0]=-(Math.PI);;
dw=2.0*Math.PI/((float)M1);
for(I=1;I<Nt;I++)
        W[I]=W[I-1]+dw;

for(I=0;I<Nt;I++)
        F_File.writeDouble(W[I]);
for(I=0;I<Nt;I++)
        F_File.writeDouble(W[I]);
for(I=0;I<Nt;I++)
        F_File.writeDouble(W[I]);


for(m=0;m<Nt;m++)
{
for(n=0;n<Nt;n++)
for(l=0;l<Nt;l++)
{
zrt=0.0;zit=0.0;

for(i=-N1;i<=N1;i++)
for(j=-N1;j<=N1;j++)
for(k=-N1;k<=N1;k++)
{
        xrm=Math.cos((double)(i*W[m]));
        xim=Math.sin(-(double)(i*W[m]));

        xrn=Math.cos((double)(j*W[n]));
        xin=Math.sin(-(double)(j*W[n]));

        xrl=Math.cos((double)(k*W[l]));
        xil=Math.sin(-(double)(k*W[l]));

        rl=(xrm*xrn-xim*xin);
        im=(xrm*xin+xim*xrn);


        rl=rl*xrl-im*xil;
        im=im*xil+rl*xil;
```

```
        zrt+=(h[k+N1][i+N1][j+N1])*rl;
                zit+=(h[k+N1][i+N1][j+N1])*im;
        }
                mag=Math.sqrt((double)(zrt*zrt+zit*zit));
                F_File.writeDouble(mag);
                g.drawString("Writting the Frequency response",10,300);
        }
        }
        //----------
        done=1;
        F_File.close();
        g.drawString(Fr+"    Frequency Response fie is Written",300,20);
  }
        catch(IOException e)
    {
       System.err.println("Proble in reading The File\n"+e.toString());
       System.exit(1);
       }

  }
//------------------
        private double Filter(double R2,double CF)
        {
        if    (Type==0)  return(R2/(R2+0.414*CF));
        else if (Type==1)       return((0.414*CF)/(R2+0.414*CF));
        else {
                if(Type==2)   //high pass abrupt transation
             {
                  if (R2 < CF)  return (0.0);
                  else return (1.0);
             }

          else            //low pass abrupt transation
             {
                  if (R2 < CF)  return (1.0);
                  else return (0.0);
             }
```

```
            }
        }

//-----------------------------------
private double Window(double theta,double NSQRT,int T,double alpha,int N1)
        {
        double beta,sum1,sum2,t;
        int K;
        if(T==0) return(1.0);
        else if(T==1)
                return(0.5+0.5*(Math.cos(theta*NSQRT)));
        else if(T==2)
                return(0.54+0.46*(Math.cos(theta*NSQRT)));
        else if(T==3)
return(0.42+0.5*(Math.cos(theta*NSQRT))+0.08*Math.cos(2.0*theta*NSQRT));
        else
          {
                alpha/=2.0;
                beta=(NSQRT/(float)N1)*(NSQRT/(float)N1);
                beta=alpha*Math.sqrt(Math.abs(1.0-beta));
                t=alpha;
                sum1=1.0+(t*t);
                for(K=2;K<11;K++)
                {
                 t=(1.0/(double)K)*alpha*t;
                 sum1+=t*t;

                }
                t=beta;
                sum2=1.0+(t*t);
                for(K=2;K<11;K++)
                {
                 t=(1.0/(double)K)*beta*t;
                 sum2+=t*t;
                }
                return(sum2/sum1);
        }
        //---------
        }


///************************************************
 }
```

```
//this class is to be used for generating the impulse response
// it cotain the filter type and window functions
public class FWind
{
public int Type,T;
public double CF;
public FWind(int filter_type,double cut_off,int wint)
{Type=filter_type;
 CF=cut_off;
 T=wint;
}

//------------------
public double Filter(double R2)
        {
        if    (Type==1)  return(R2/(R2+0.414*CF));        //high
        else if (Type==2)       return((CF)/(CF+0.414*R2));//low
        else {
                if(Type==3)   //low pass abrupt transation
             {
                  if (R2 < CF)  return (1.0);
                  else return (0.0);
             }

          else            //high pass abrupt transation
             {
                  if (R2 < CF)  return (0.0);
                  else return (1.0);
             }
           }
        }
//---------------------------------
public double Window(double theta,double NSQRT,double alpha,int N1)
        {
        double beta,sum1,sum2,t;
        int K;
        if(T==0) return(1.0);
        else if(T==1)
                return(0.5+0.5*(Math.cos(theta*NSQRT)));
        else if(T==2)
                return(0.54+0.46*(Math.cos(theta*NSQRT)));
        else if(T==3)
return(0.42+0.5*(Math.cos(theta*NSQRT))+0.08*Math.cos(2.0*theta*NSQRT));
```

```
        else
          {
                alpha/=2.0;
                beta=(NSQRT/(float)N1)*(NSQRT/(float)N1);
                beta=alpha*Math.sqrt(Math.abs(1.0-beta));
                t=alpha;
                sum1=1.0+(t*t);
                for(K=2;K<11;K++)
                {
                 t=(1.0/(double)K)*alpha*t;
                 sum1+=t*t;

                }
                t=beta;
                sum2=1.0+(t*t);
                for(K=2;K<11;K++)
                {
                 t=(1.0/(double)K)*beta*t;
                 sum2+=t*t;
                }
                return(sum2/sum1);
        }
//---------
}
}
```

APPENDIX C

DESIGN OF 3-D IIR FILTERS USING SHANK'S METHOD COMPUTER PROGRAMS

```
//applet for generating the impulse response of IIR Filters
import java.applet.*;
import java.awt.*;
import java.io.*;
import wingFrame;

public class WIIRI3 extends Applet
{
        RandomAccessFile I_File,F_File;
        boolean m_fStandAlone = false;
        public double h[][][];
        public double hr[][][],hi[][][];
        public FFT3DC F;
        public FWind W;



        public static void main(String args[])
        {
                wingFrame frame = new wingFrame("wingI");
                frame.show();
    frame.hide();
                frame.resize(frame.insets().left + frame.insets().right  + 600,
                                        frame.insets().top  + frame.insets().bottom + 400);

                WIIRI3 applet_wing = new WIIRI3();

                frame.add("Center", applet_wing);
                applet_wing.m_fStandAlone = true;
                applet_wing.init();
                applet_wing.start();
    frame.show();
        }
  //--------------------
        public WIIRI3()
        {
        }

        //--------------------

        public void init()
```

```
    {
            h=new double[32][32][32];
            hr=new double[32][32][32];
            hi=new double[32][32][32];
       resize(600, 400);

    }
//---------------------
    public void paint(Graphics g)
    {

    int M,M1,M2,I,J,K,II,JJ,KK,Order=3,Type;
    double theta,NSQRT,R2,CF=1.4,NSQ;
    int WT=0,L1;
    M1=32;
    M=M1>>1;
    Type=2;
    CF=(CF/Math.PI)*M;
    CF*=CF;

    if(Type<3)
    {for(I=0;I<Order;I++)
     CF*=CF;
    }

    W=new FWind(Type,CF,WT);



    for(I=0;I<M1;I++)
    {
    II=(I-M)*(I-M);
    for(J=0;J<M1;J++)
    {
    JJ=(J-M)*(J-M);
    for(K=0;K<M1;K++)
    {
    KK=(K-M)*(K-M);
    R2=(double)(JJ+II+KK);
    if(Type < 3)
    {for(L1=0;L1<Order;L1++)
     R2*=R2;
    }
```

```
        hr[I][J][K]=W.Filter(R2)*(Math.pow(-1,(I+J+K)));
        hi[I][J][K]=0.0;
        }
        }
        }

        F=new FFT3DC(hr,hi,M1,1);

        //----------
        M2=M1/2;
        for(I=0;I<M1;I++)
                for(J=0;J<M1;J++)
                  for(K=0;K<M1;K++)
             h[I][J][K]=(F.FFT3D_Out(I,J,K,0)*Math.pow(-1,I+J+K));


//-----------------------------------------//
        theta=(Math.PI)/(M2*Math.sqrt((double)3.0));
        for(I=0;I<M1;I++)
        {
        II=(I-M2)*(I-M2);
        for(J=0;J<M1;J++)
        {JJ=(J-M2)*(J-M2);
        for(K=0;K<M1;K++)
        {
         KK=(K-M2)*(K-M2);
         NSQ=(double)(II+JJ+KK);
         NSQRT=Math.sqrt((double)NSQ) /(Math.sqrt((double)3.0));
         h[I][J][K]*=W.Window(theta,NSQRT,1,M2);
        }
        }
        }


        //------------
try
        {
        I_File  =new RandomAccessFile      (("c:/idris/Data/IIP.dat"),"rw");

        I_File.writeInt(M1);
        I_File.writeInt(M1);
        I_File.writeInt(M1);
        for(I=0;I<M1;I++)
```

```
        for(J=0;J<M1;J++)
        for(K=0;K<M1;K++)
           I_File.writeDouble(h[I][J][K]);
           I_File.close();


        }
        catch(IOException e)
  {
        System.err.println("Proble in reading The File\n"+e.toString());
        System.exit(1);
  }


  g.drawString("Done Writting the impulse response",100,30);

}


}//----------end of the applet----------
```

```
//this applet is for obtaining
//the filter coefficients by solving
//algebraic equations and generating
//the frequency response of the IIR filter
//
import java.applet.*;
import java.awt.*;
import java.io.*;
import wingFrame;

public class IIF3 extends Applet
{
        RandomAccessFile I_File,F_File,X_File,P_File;
        boolean m_fStandAlone = false;
        public double h[][][];
        public double hr[][][],hi[][][],a[][][],b[][][],A[][];
        public int Type,M,I,J,K,NU=0;

public static void main(String args[])
        {
                wingFrame frame = new wingFrame("wing");
                frame.show();
      frame.hide();
                frame.resize(frame.insets().left + frame.insets().right  + 600,
                                        frame.insets().top  + frame.insets().bottom + 400);

                IIF3 applet_wing = new IIF3();

                frame.add("Center", applet_wing);
                applet_wing.m_fStandAlone = true;
                applet_wing.init();
                applet_wing.start();
      frame.show();
        }
   //--------------------
        public IIF3()
        {

        }

        //--------------------
public void init()
        {
```

```
   resize(600, 400);
try
        {
        I_File  =new RandomAccessFile     (("c:/idris/data/IIP.dat"),"r");
        F_File  =new RandomAccessFile     (("c:/idris/data/FRS.dat"),"rw");
        X_File  =new RandomAccessFile     (("c:/idris/data/XRS.dat"),"rw");
        P_File  =new RandomAccessFile     (("c:/idris/data/PHS.dat"),"rw");

        M=I_File.readInt();
        M=I_File.readInt();
        M=I_File.readInt();

        h=new double[M][M][M];
        hr=new double[M][M][M];
        hi=new double[M][M][M];
        for(I=0;I<M;I++)
         for(J=0;J<M;J++)
           for(K=0;K<M;K++)
            hr[I][J][K]=I_File.readDouble();
            I_File.close();
        }

catch(IOException e)
   {
        System.err.println("Proble in reading The File\n"+e.toString());
        System.exit(1);
   }
}
   //--------------------
        public void paint(Graphics g)
        {
        int      N=2,M1=0,M2=0,NO=N-1,N1,N2,Row,Col,i,j,m,n,l,k,ypos=10,xpos=10;
        int Nt,KK,LL,MM;
        double sum,ASUM,zrb,zib,xrl,xil;
        double W[],xrm,xim,xrn,xin,zrt,mag,Phase,zit,dw,rl,im,ZX;
        M1=(M/2)+NO;
        M2=(M/2)-NO;

        N1=(N+1)*(N+1)*(N+1)-1;
        N2=(N+1)*(N+1)*(N+1);
        a=new double [N+1][N+1][N+1];
        b=new double [N+1][N+1][N+1];
        A=new double [N1][N2];
```

```
for(I=M2;I<M;I++)
        for(J=M2;J<M;J++)
      for(K=M2;K<M;K++)
       h[I-M2][J-M2][K-M2]=hr[I][J][K];



Row=0;
for(MM=0;MM<=N;MM++)
 for(LL=0;LL<=N;LL++)
   for(KK=0;KK<=N;KK++)
    {
if((MM+LL+KK) ==0)continue;
          Col=0;
          for(i=0;i<=N;i++)
          for(j=0;j<=N;j++)
          for(k=0;k<=N;k++)
          {
          if((i+j+k) ==0)continue;
          if( Col>= Row)
          {
          sum=0.0;
          for(m=0;m<M1;m++)
                 for(n=0;n<M1;n++)
                   for(l=0;l<M1;l++)
              {
                   if(n <= N && m<=N && l <=N) continue;
                   if(((m-i)<0) || ((n-j) <0) || ((m-MM) <0) || ((n-LL)<0)  || ((l-k) < 0) ||
((l-KK) < 0) ) continue;
                   sum+=h[(m-i)][(n-j)][l-k]*h[(m-MM)][(n-LL)][l-KK];

                   }
          A[Row][Col]=sum;
                 }

          else  A[Row][Col] =A[Col][Row];
           Col++;
           }

           sum=0.0;
           for(m=0;m<M1;m++)
```

```
            for(n=0;n<M1;n++)
            for(l=0;l<M1;l++)
            {
            if(n<=N && m<= N && l<=N) continue;
            if((m-MM)<0 || (n-LL) <0  || ((l-KK)<0))continue;
            sum-=h[m][n][l]*h[(m-MM)][(n-LL)][l-KK];
            }
            A[Row][N1]=sum;
            Row++;
    }


SIMQ(A,N1,g);
if(NU==1)
{
g.drawString("No unique Solution ",100,10);
System.exit(0);
}


b[0][0][0]=1.0;
Row=0;
for(I=0;I<=N;I++)
  for(J=0;J<=N;J++)
          for(K=0;K<=N;K++)
      {if((I+J+K)==0)continue;
      b[I][J][K]=A[Row][N1];
                  Row++;
      }
  for(m=0;m<=N;m++)
          for(n=0;n<=N;n++)
           for(l=0;l<=N;l++)
            {
            a[m][n][l]=0.0;
           for(I=0;I<=N;I++)
            for(J=0;J<=N;J++)
             for(K=0;K<=N;K++)
            {
   if((m-I)<0 || (n-J)<0 || (l-K)<0) continue;
   a[m][n][l]+=b[I][J][K]*h[(m-I)][(n-J)][(l-K)];
            }
     }
```

```
xpos=10,ypos=10;
        g.drawString("The a Coefficients",100,ypos);
        g.drawString("The b Coefficients",300,ypos);
        ypos=20;
        for(I=0;I<N+1;I++)
        for(J=0;J<N+1;J++)
        for(K=0;K<N+1;K++)
        {
        g.drawString("a["+I+","+J+","+K+"]=  "+a[I][J][K] , 100 , ypos);
        g.drawString("b["+I+","+J+","+K+"]=  "+b[I][J][K] , 300 , ypos);
        ypos+=10;


        }


//----------------------
//calculating and storing the impulse response

h=new double[M][M][M];
for(m=0;m<M;m++)
for(n=0;n<M;n++)
for(l=0;l<M;l++)
{
if(m<=N && n <=N && l <=N)
h[m][n][l]=a[m][n][l];
else h[m][n][l]=0.0;
for(I=0;I<=N;I++)
for(J=0;J<=N;J++)
for(K=0;K<=N;K++)
{if((I+J+K)==0) continue;
if((m-I)<0 || (n-J) < 0 || (l-K) <0) continue;
h[m][n][l]-=b[I][J][K]*h[m-I][n-J][l-K];
}
}


try
{
X_File.writeInt(M);
X_File.writeInt(M);
X_File.writeInt(M);
```

```
for(m=0;m<M;m++)
for(n=0;n<M;n++)
for(l=0;l<M;l++)
X_File.writeDouble(h[m][n][l]);
X_File.close();
g.drawString("Done Writting the impulse response",10,290);
}
catch(IOException e)
 {
  System.err.println("Proble in reading The File\n"+e.toString());
  System.exit(1);
 }



//----------------------

try
        {
        Nt=M+1;
        F_File.writeInt(Nt);
        F_File.writeInt(Nt);
        F_File.writeInt(Nt);


        P_File.writeInt(Nt);
        P_File.writeInt(Nt);
        P_File.writeInt(Nt);

        W=new double[Nt];
        W[0]=-(Math.PI);;
        dw=2.0*Math.PI/((double)M);
        for(I=1;I<Nt;I++)
                W[I]=W[I-1]+dw;

        for(I=0;I<Nt;I++)
        {
                F_File.writeDouble(W[I]);
                P_File.writeDouble(W[I]);}
        for(I=0;I<Nt;I++)
        {
                F_File.writeDouble(W[I]);
                P_File.writeDouble(W[I]);
```

```
    }
for(I=0;I<Nt;I++)
    {       F_File.writeDouble(W[I]);
            P_File.writeDouble(W[I]);
    }
    for(m=0;m<Nt;m++)
    for(n=0;n<Nt;n++)
    for(l=0;l<Nt;l++)
    {
    zrt=(double)0.0;zit=(double)0.0;
    zrb=(double)0.0;zib=(double)0.0;

    for(i=0;i<=N;i++)
    for(j=0;j<=N;j++)
    for(k=0;k<=N;k++)
    {
            xrm=Math.cos((double)(i*W[m]));
            xim=Math.sin(-(double)(i*W[m]));

            xrn=Math.cos((double)(j*W[n]));
            xin=Math.sin(-(double)(j*W[n]));

            xrl=Math.cos((double)(k*W[l]));
            xil=Math.sin(-(double)(k*W[l]));

            rl=(xrm*xrn-xim*xin);
            im=(xrm*xin+xim*xrn);

            rl=rl*xrl-im*xil;
            im=im*xrl+rl*xil;

    zrt+=a[i][j][k]*rl;
            zit+=a[i][j][k]*im;

            zrb+=b[i][j][k]*rl;
            zib+=b[i][j][k]*im;
    }

           mag=(Math.sqrt((double)(zrt*zrt+zit*zit)));
           mag/=(Math.sqrt((double)(zrb*zrb+zib*zib)));

           Phase=Math.atan2(zrt,zit)-Math.atan2(zrb,zib);
```

```
                F_File.writeDouble(mag);
                P_File.writeDouble(Phase);
        }

        F_File.close();
        P_File.close();
        g.drawString("Done Writting the Frequency response",10,300);
   }
        catch(IOException e)
     {
          System.err.println("Proble in reading The File\n"+e.toString());
          System.exit(1);
         }


  }

//*************************************************
public void SIMQ(double X[][],int N,Graphics g)
        {
        int I,J,K,L,N1;
        double Pivt,temp;


        N1=N+1;
        for(J=0;J<N;J++)
        {
        Pivt=Math.abs(X[J][J]);
        L=J;
        for(K=J;K<N;K++)
        {
         if(Pivt <Math.abs(X[K][J]));
         {
         Pivt=Math.abs(X[K][J]);
         L=K;
         }
        }
        if(Pivt <1.e-10)
        {NU=1;
        return;
        }

        for(K=0;K<N1;K++)
```

```
        {
        temp=X[J][K];
        X[J][K]=X[L][K];
        X[L][K]=temp;
        }

        for(K=J+1;K<N1;K++)
        X[J][K]/=X[J][J];
        X[J][J]=1.0;
        for(I=0;I<N;I++)
         {
         if(I !=J)
          {
            for(K=J+1;K<N1;K++)
                  X[I][K]-=X[I][J]*X[J][K];
             X[I][J]=0.0;
          }
         }
        }
        return;
}


///***********************************************

}//----------end of the applet----------
```

VITA AUCTORIS

Idris S. El-Feghi

1959 Born on March 28th, Misurata-Libya

1977 High School Diploma in Mathematics and Applied Science, Misurata Secondary School, Misurata-Libya.

1983 B.Sc.. Electrical Engineering, University of Portland , Portland, Oregon, USA.

83-88 Tajura Research Center as a design engineer.

88-93 General Secretariate of Industry as a manager of Industrial Information Center.

93-95 Misurata high polytechnic institute.

1999 Candidate for the degree of Master of Applied Science in Electrical and Computer Engineering at the University of Windsor, Windsor, Ontario, Canada.